Optimisation of Surface Mount Device Placement Machine

in

Printed Circuit Board Assembly

by

Masri Ayob

MEng (UTMalaysia), BEng (UKMalaysia)

A thesis submitted to the University of Nottingham for the degree of Doctor of Philosophy

in

The School of Computer Science and Information Technology

June 2005

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

Avg	: Average.
CDPP	: Chebychev Dynamic Pick-and-Place Point.
CPU	: Central Processing Unit (of a computer/machine).
CT	: Total Assembly Cycle Time i.e. the time elapsed between two completed PCBs.
CT_{avg}	: Average CT.
CT_o	: Initial CT.
CX	: Cycle crossover.
EDPP	: Extended Dynamic Pick-and-Place Point.
EMC	: Exponential Monte Carlo.
EMCQ	: Exponential Monte Carlo with Counter.
DPP	: Dynamic Pick-and-Place Point.
FPP	: Fixed Pick-and-Place Point.
GA	: Genetic Algorithm.
HSCS	: High Speed Chip Shooter.
I	: Improvement.
ISR	: Interrupt Service Routine.
KB	: Knowledge Based.
LLH	: Low-level Heuristic.
LS	: Local Search.
MC	: Monte Carlo.
mm	: millimetre.
ms	: millisecond.
NP	: Non-deterministic polynomial.
NP-hard	: The NP class.
P	: Polynomial.
PC	: Personal Computer.
PCB	: Printed Circuit Board.
PCBA	: Printed Circuit Board Assembly.

PCTSP	: Precedence Constrained Travelling Salesman Problem.
PMX	: Partially Mapped crossover.
QAP	: Quadratic Assignment Problem.
SA	: Simulated Annealing.
SCARA	: Single Compliance Robot for Assembling.
SD	: Steepest Descent.
SMD	: Surface Mount Device.
SMT	: Surface Mount Technology.
TS	: Tabu Search.
TSP	: Travelling Salesman Problem.
VNMS	: Variable Neighbourhood Monte Carlo Search.
VNS	: Variable Neighbourhood Search.
VRP	: Vehicle Routing Problem.

ABSTRACT

Optimisation of surface mount device placement machines is a challenging optimisation problem with many opportunities to improve the efficiency of the machine. When hundreds or thousands of electronic components, of different shapes and sizes, have to be placed at specific locations on a printed circuit board, finding an optimal assembly operation is complicated and time consuming.

The ultimate goal of this thesis is to produce an effective scheduling approach that can enhance the throughput of a surface mount device placement machine. Towards this goal, the research is more focused on improving the pick-and-place operation. The research also focuses on enhancing the robot motion control, nozzle selection and feeder setup. Since the optimisation of the surface mount device placement machine is machine specific, this work is concerned with multi-head and sequential pick-and-place surface mount device placement machines.

The research begins by investigating the operational methods of various placement machines, proposes five categories of machines based on their specification and operational methods and attempts to relate the heuristics to the machine types and identifies the scheduling issues related to the machine types. Next, the research proposes a revised dynamic pick-and-place point specification approach based on chebychev distance (i.e. this distance is calculated as the maximum of the Y or X coordinate). This work introduces a triple objective function that aims to minimise the assembly cycle time and the movement of the feeder carrier and PCB table. The research also proposes a methodology for on-line scheduling, a Monte Carlo based acceptance criteria, a Monte Carlo based hyper-heuristic, a Variable Neighbourhood Monte Carlo Search and a novel weighted nozzle rank heuristic to optimise the component pick-and-place operation (and/or nozzle optimisation) of multi-head surface mount device placement machines. Computational results are presented to demonstrate the effectiveness of these approaches.

ACKNOWLEDGEMENTS

I would like to thank my academic supervisor, Dr Graham Kendall, for his encouragement, kindness, support, patience and valuable guidance throughout this study. I am also grateful for the many opportunities for attending and presenting conference papers.

I started my PhD with Prof. Peter Cowling (my ex-supervisor). Many thanks to Prof. Peter Cowling for his help and encouragement.

I wish to express my heartfelt thanks to my husband, Mazlan, who has been the source of my inspiration, and my lovely sons, Afifi, Aiman and Akmal, for giving me the courage to complete my PhD study.

To my parents, my sister and my brother, many thanks for your constant support and understanding.

Many thanks to my colleagues in the ASAP Research group, especially Djamila Ouelhadj, Eric Soubeiga, Steve and all the Malaysians (Aniza, Baharuddin, Hisham, Naimah, Razali and Salwani), for making my time spent at the University of Nottingham an enjoyable experience.

To Prof. Edmund Burke, the ASAP Research Group leader, many thanks for his help, support and professionalism.

My research benefited from collaborations with Paul Klomp and Geert Dortmans from DIMA SMT Systems.

To my examiners, Dr. Jon Garibaldi and Dr. Paul Stewart, many thanks for your valuable comments, which have improved the contents of my thesis.

Last but not least, it is my pleasure to acknowledge Public Services Department of Malaysia (financer), National University of Malaysia (financer) and DIMA SMT Systems (our company collaborator).

Many thanks to you all and may God bless us.

To Mazlan, Afifi, Aiman, Akmal

and

My Parents

DECLARATION

I hereby declare that this thesis has not been submitted, either in the same or different form, to this or any other university for a degree.

signature:

Chapter 1

Introduction

1.1 Background and Motivation

Over the last 20 years, PCB (printed circuit board) production has evolved from a labour-intensive activity to a highly automated activity (Crama et al., 1997). Surface Mount Technology (SMT) has almost replaced pin-through-hole technology for PCB assembly and has enabled the production of high density (allowing many components to be placed onto a PCB in a small area) PCBs (Jeevan et al., 2002). To be more competitive in today's global marketplace, PCB assembly manufacturers are striving to respond to emerging trends including high quality, low-cost and just in-time delivery. Therefore, in order to enhance their competitiveness, many PCB assembly manufacturers are keen to develop a computer integrated manufacturing system that is capable of producing an effective planning, scheduling and control procedure. Moreover, the niche of automating the printed circuit board assembly (PCBA) is increasing with the miniaturisation of component designs and the increasing density of components on the PCB (Moyer and Gupta, 1996a, 1996b).

Generally, an SMT assembly line involves solder paste, component placement and solder reflow operations (a soldering process to adhere components on the PCB) (Tirpak, 2000). An SMD (surface mount device) placement machine is very expensive (US$300,000 to US$1,000,000) and yet the SMT lines are typically designed such that the SMD placement machine is the limiting resource or “bottleneck”, which is the key issue for assembly line optimisation (Csaszar et al., 2000a; Moyer and Gupta, 1997; Tirpak et al., 2000).

Typically, the placement operation begins by loading the PCB into the SMD placement machine (e.g. via a conveyer system). Next, a “fiducial marks” operation is performed to identify the exact position and orientation of the PCB inside the SMD placement machine. The “fiducial marks” are special points (typically 2-4 points) that are usually located at the corners of the PCB (Magyar et al., 1999). Then, the components are assembled onto the PCB guided by the optimisation software that has been installed in the SMD placement machine. Finally, once completed (or partially completed, e.g. due to component run out or job completion), the PCB is moved out from the SMD placement machine. Before undergoing the solder reflow operation, the components are secured onto the PCB by using adhesive or solder paste (Leu et al., 1993).

In practice, the SMD placement machines are usually not equipped with efficient optimisation software (Shih et al., 1996). Until now, the PCB machine vendors and software companies have not been capable of solving even a single machine problem efficiently (Magyar et al., 1999). This indicates the need for research in this area.

In the early 1980s, the first pick-and-place SMD placement machine, with only one placement head, was introduced (Bentzen, 2000). Nowadays, there are many types of SMD placement machines available, such as sequential pick-and-place, rotary disk turret, concurrent pick-and-place, etc. (Gastel, 2002; Grotzinger, 1992; Khoo and Loh, 2000). Various types of SMD placement machines have different characteristics and restrictions (Wang et al., 1999). Thus, the PCB production scheduling process is highly influenced by the type of SMD placement machine being used (Burke et al., 2001).

Owing to the lack of standardisation among SMD placement machines, the optimisation of the pick-and-place operations in printed circuit board assembly line is mainly influenced by the constraints of the SMD placement machine and the characteristics of the production environment (Duman and Or, 2004; Leipälä and Nevalainen, 1989; Shih et al., 1996).

Generally, each placement machine has a feeder carrier (or feeder magazine), PCB table, head, nozzle (tool or gripper) and a tool magazine. The feeder carrier, PCB table and head can be either fixed or moveable depending on the specification of the machine. In some cases, the feeder carrier is divided into different feeder banks, each consisting of feeder slots (Wang et al., 1999). Each feeder bank consists of several feeder slots where the component feeders are located. The feeders are used to provide the machine with a continuous supply of components.

Optimisation in PCB assembly involves a list of sub-problems which have to be addressed, such as an assignment of PCB types to product families and to machine groups, allocation of components to machines and location of components in feeder slots and component placement sequencing (Crama et al., 2002). These sub-problems are tightly intertwined and practically impossible to solve to optimality. These pose a great optimisation challenge, which needs to be explored.

1.2 Scope

The optimisation of the SMD placement machine problem is chosen as it poses a great optimisation challenge with the potential to improve machine efficiency and there is always the possibility of developing commercial links with industry to develop the work even further.

Since there are various types of SMD placement machines, all which have different characteristics and restrictions, the component pick-and-place scheduling is highly influenced by the type of SMD placement machine being used. Therefore, a decision has to be made as to which type of SMD placement machine the research should be conducted upon. We have decided to focus this research on optimising the multi-head SMD placement machine (refer to 3.6.4). However, before clearly understanding the modus operandi of the multi-head machine, the early stage of the research focuses on the sequential pick-and-place SMD placement machine (refer to 3.6.5) since the operational methods of the machine are easy to understand and the optimisation factors of the machine are not as complicated as other machine types.

Various optimisation problems exist in the area of production planning for the assembly of PCBs. These being assigning PCB types to product families and to machine groups, allocating component feeders to machines, assigning component feeders to slots on the feeder carrier (feeder setup), sequencing the component pick-and-place operations etc. The dilemma is that all the sub problems are tightly intertwined and probably NP-hard (Garey and Johnson, 1979; Truss, 1999), and the question arises as to which one should be solved first. As a consequence, this research is more focused on optimising the component pick-and-place sequence by assuming that other optimisation problems are solved. The research is concerned with a single PCB type of a single SMD placement machine problem. The research also focuses on improving the robot motion control and feeder setup.

In order to solve the real world SMD placement machine problem, the research also uses a hybrid pick-and-place machine (specifically a new DIMA machine called Hybrid P&P HP-110), which is a type of multi-head SMD placement machine.

1.3 Objectives

The ultimate goal of this thesis is to produce an effective scheduling approach that can enhance the SMD placement machine throughput. To accomplish this aim, we identified the following objectives:

a) To classify the SMD placement machines based on their specification and operational methods, and then associate the machine categories with the models and heuristics.

b) To design a new DPP (dynamic pick-and-place) approach, which we call CDPP (chebychev dynamic pick-and-place). The CDPP approach is modelled in such a manner that individual mechanism delays are minimised. The approach focuses on improving the robot motion control.

c) To develop an on-line scheduling approach for optimising the pick-and-place sequence of an SMD placement machine.

d) To design a new acceptance criterion based on a Monte Carlo approach.

e) To apply hyper-heuristic and variable neighbourhood approaches for solving the pick-and-place sequence of an SMD placement machine problem.

f) To solve the pick-and-place sequencing problem of a real-world machine.

1.4 Contributions

The work carried out in this thesis has contributed in the following areas:

a) A Monte Carlo based acceptance criteria has been designed, which has led to the introduction of a Monte Carlo based hyper-heuristic. This has further improved the optimisation of the multi- head SMD placement machine. The proposed acceptance criteria, Exponential Monte Carlo (EMC) and Exponential Monte Carlo with Counter (EMCQ) has subsequently been successfully adopted by Abdullah et al. (2004) and Kendall and Mohamad (2004a) to solve examination timetabling and frequency assignment problems, respectively.

b) A new heuristic, which is a revised dynamic pick-and-place point (DPP) specification approach called Chebychev DPP (CDPP) and a triple objective function that attempts to minimise the assembly cycle time together with the minimisation of the movement of the feeder carrier and the PCB table, has been developed.

c) A Variable Neighbourhood Monte Carlo Search (VNMS) heuristic, which employs variable neighbourhood search with an Exponential Monte Carlo acceptance criterion, has been developed.

d) An on-line constructive heuristic and a novel weighted nozzle rank heuristic to optimise the component pick-and-place operations of the hybrid pick-and-place machine of a new SMD placement machine has been presented.

e) A methodology for on-line scheduling to sequence the pickup-and-placement of components on a multi-head SMD placement machine has been proposed.

1.5 Dissemination

The research work carried out for this thesis has been disseminated in international journals and international refereed conference proceedings. The following is a list of published and submitted papers arising from the work reported in this thesis.

1.5.1 Conference Papers

a) Ayob, M., Cowling, P. and Kendall, G.(2002) Optimisation for surface mount placement machines. *Proceedings of the IEEE ICIT'02*, Bangkok, 11-14 Dec, 498-503. (This work is reported in chapter 3 of this thesis).

b) Ayob, M. and Kendall, G. (2002). A new dynamic point specification approach to optimise surface mount placement machine in printed circuit board assembly. *Proceedings of the IEEE ICIT'02*, Bangkok, 11-14 Dec, 486-491. (This work is reported in chapter 4 of this thesis).

c) Ayob, M. and Kendall, G. (2003a). Real-time scheduling for multi headed placement machine. *Proceedings of the 5th IEEE International Symposium on Assembly and Task Planning, ISATP'03*, Besançom, France, 9-11 July, 128-133. (This work is reported in chapter 5 of this thesis).

d) Ayob M. and Kendall G. (2003b). An investigation of an adaptive scheduling for multi headed placement machines. *Proceedings of the 1st Multidisciplinary International Conference on Scheduling: Theory and Applications, MISTA 2003*, Nottingham, UK, 13-16 Aug, 363-380. (This work is reported in chapter 6 of this thesis).

e) Ayob, M. and Kendall, G. (2003c). A Monte Carlo hyper-heuristic to optimise component placement sequencing for multi head placement machine. *Proceedings of the International Conference on Intelligent Technologies, InTech'03*, Chiang Mai, Thailand, 17-19 Dec, 132-141. (This work is reported in chapter 6 of this thesis).

f) Ayob, M. and Kendall, G. (2004). A nozzle selection heuristic to optimise the hybrid pick and place machine. *Proceedings of the 2004 IEEE*

Conference on Cybernetics and Intelligent Systems (CIS 2004), Singapore, 1259-1264. (This work is reported in chapter 8 of this thesis).

1.5.2 Journal Papers

a) Ayob M. and Kendall G. (2005a). A triple objective function with a chebychev dynamic point specification approach to optimise the SMD placement machine. *European Journal of Operational Research,* 164, 609-626. (This work is reported in chapter 4 of this thesis).

b) Ayob M. and Kendall G. (2005b). A Variable Neighbourhood Monte Carlo Search For Component Placement Sequencing Of Multi Headed Placement Machine in Printed Circuit Board Assembly. Submitted to the *Intelligent Manufacturing Journal*. (This work is reported in chapter 7 of this thesis).

c) Ayob M. and Kendall G. (2005c). A Weighted Nozzle Rank Heuristic to Optimise the Hybrid Pick and Place Machine. Submitted to the *International Journal of Production Research*. (This work is reported in chapter 8 of this thesis).

1.6 Thesis outline

The thesis is organised into nine chapters:

Chapter 2 discusses the complexity of the printed circuit board assembly (PCBA) optimisation problem, particularly the pick-and-place sequencing problem. Some common terms that are usually used in PCBA and some standard heuristics/meta-heuristics that are commonly applied in solving the pick-and-place sequencing problem are discussed.

Chapter 3 reviews a single SMD placement machine optimisation in PCB assembly. Due to insufficient information in some of the reported works, it only covers the papers that provide high-level descriptions of the algorithms, implementations and summaries of experimental results. Indeed, many papers usually do not provide enough information, which does not allow other researchers to make comparisons or evaluate their own work. Moreover, there are various types of SMD placement machines that have different characteristics and restrictions, which highly influence the production scheduling approach. The chapter proposes five categories of SMD placement machines based on their specification and operational methods; these being dual delivery, multi-station, turret-type, multi-head and sequential pick-and-place SMD placement machines. It attempts to associate the heuristics to the machine types and identify the scheduling issues related to the machine types.

Chapter 4 describes the techniques for improving robot motion control and reviews the related work. Since robot motion control is more applicable to a sequential pick-and-place SMD placement machine, the work in this chapter only focuses on a problem of a sequential pick-and-place SMD placement machine that has a single arm with a single head equipped with a single nozzle. The chapter proposes a revised dynamic pick-and-place point (DPP) specification approach called Chebychev DPP (CDPP). A triple objective function that attempts to minimise the assembly cycle time together with the minimisation of the movement of the feeder carrier and the PCB table is also presented. Experimental results are included to evaluate the proposed approaches.

Chapter 5 proposes a methodology for on-line scheduling to sequence the pickup-and-placement of component on a multi-head SMD placement machine in printed circuit board assembly (PCBA) to overcome spontaneous circumstances. The on-line scheduling algorithm can eliminate the machine's idling time by starting the pickup and placement operations immediately after the PCB (and the PCB data) have been loaded into the machine and the machine can continuously run even if there are missing components or a feeder changeover occurs. While the placement machine is assembling components, the scheduler might employ free CPU time (whilst the robot arm is moving) to improve the initial schedule by using a random descent search technique. Thus, subsequent PCBs will use the improved schedule. The factors involved in determining the efficiency of pick-and-place operations of multi-head SMD placement machines such as the grouping of PCB points (also referred to as placement points) to a sub tour, pipette/nozzle assignment, pickup-and-placement sequencing etc. are discussed in detail. Again, experimental results are reported to demonstrate the effectiveness of the approach.

Chapter 6 studies a hyper-heuristic approach to further improve a multi-head SMD placement machine. It reviews some hyper-heuristic approaches and then develops two constructive heuristics; a randomised and an ordered constructive heuristic. Three types of acceptance criteria based on a Monte Carlo approach are introduced. These being a Linear Monte Carlo (LMC), an Exponential Monte Carlo (EMC) and an Exponential Monte Carlo with counter (EMCQ). Based on these acceptance criteria, a Monte Carlo based hyper-heuristic is proposed to sequence the pick-and-place operations of the multi-head SMD placement machine. Experimental results are presented for evaluation purposes.

Chapter 7 focuses on a VNS (Variable Neighbourhood Search) approach for optimising the component pick-and-place sequence of a multi-head SMD placement machine. A Variable Neighbourhood Monte Carlo Search (VNMS), that employs a variable neighbourhood search technique with an Exponential Monte Carlo acceptance criterion is introduced. Experimental results are included to evaluate the proposed approaches.

Chapter 8 is devoted to modelling a real world SMD placement machine that is the hybrid pick-and-place machine (specifically a new DIMA machine called Hybrid P&P HP-110), a multi-head SMD placement machine. The chapter presents an on-line constructive heuristic that gives the highest priority to minimising the number of nozzle changes, then maximising simultaneous vision operations, simultaneous pickups and same feeder bank pickups. The on-line scheduling approach utilises a greedy search that can concurrently generate a schedule for the subsequent PCB points using spare CPU time during pick-and-place operations. This approach is different from the work in chapter 4, which only allowed the machine to begin a pickup and placement operation once a complete schedule was available, prepared an improved schedule for the subsequent PCB only (not for the current PCB being processed), ignored the nozzle change operation (it assumed that all components can be picked up by the same nozzle) and modelled different types of machine specification. A novel weighted nozzle rank heuristic for optimising the machine type is also presented in this chapter. Experimental results are included to demonstrate the effectiveness of the proposed approaches.

Finally, chapter 9 summarises the research. It discusses the contributions in this work and then proposes further research directions that may be undertaken.

Chapter 2

Optimisation Issues in Printed Circuit Board Assembly

2.1 Introduction

This chapter is concerned with the complexity of the optimisation issues in printed circuit board assembly (PCBA), particularly in component pick-and-place scheduling. We initially describe some of the common terms used in PCBA. The chapter also discusses some of the standard heuristics/meta-heuristics that are commonly used in many optimisation problems that are also applicable in solving the surface mount device (SMD) placement machine optimisation problem. It is not intended to be an exhaustive survey of the field but aims to provide a consistent background of meta-heuristics, which will underpin the rest of the thesis.

2.2 Common Terms in Printed Circuit Board Assembly

Arm	:	Holds the head(s) and is used to move the head(s) for pick-and-place operations.
Component feeder	:	Is used to provide the machine with a continuous supply of components.
Feeder bank	:	Consists of several feeder slots where the component feeders are located.

Feeder carrier	:	Is mounted on one, two, three, or four sides of the machine that holds the feeder banks.
Head	:	Is usually located at the end of the arm and is movable to transport the components between feeder carrier and PCB on the PCB table. It equipped with pipette(s).
Nozzle	:	Sometimes referred to as tool or gripper and is used to grasp the components.
PCB table	:	Is used to hold the PCB in a locked position during a pick-place operation.
Pipette	:	Sometimes referred to as spindle and is located at the head and used to hold a nozzle. It can move in *Z* direction (up-down) to perform pick-and-place operations.
Tool bank	:	Stores a number of nozzles (tools) where the nozzles can be changed automatically from a tool bank, as necessary. Sometimes, it is referred to as a tool magazine.

2.3 Printed Circuit Board Assembly Issues

Electronic components (possibly hundreds or thousands) are assembled onto a PCB (printed circuit board) using an SMD (surface mount device) placement machine. The optimisation of the feeder setup and component pick-and-place sequence, are very important for the efficiency of SMD placement machines. When hundreds of electronic components of different shapes and sizes have to be placed at specific locations on a printed circuit board (PCB), finding an optimal robot travelling route is complicated and time consuming (Su and Fu, 1998). In general, the component pick-and-place sequencing problem is modelled as a travelling salesman problem (TSP), which is a strongly NP-hard problem. Hence, this problem is also a strongly NP-Hard optimisation problem and most practical instances are difficult to solve to optimality in a reasonable time (Ellis et al., 2001; De Souza and Lijun, 1995). Indeed, the general PCB assembly problem is at least as complex as the TSP, which is known to be NP-complete (Nelson and Wille, 1995).

The PCBA problem is easy to describe, but due to the NP characteristics of the sub problems involved, it is practically impossible to solve instances to optimality by mathematical programming approaches due to a heavy computational burden (Moyer and Gupta, 1996b). An exact solution using optimisation theory is unrealistic (Nelson and Wille, 1995). For example, the component pick-and-place sequencing problem is a quadratic integer program (IP) that is difficult to solve using exact methods for even unrealistically small problems (Liggertt, 1981). The complexity of the problem is due to the interrelated sub problems where the quality of the component pick-and-place sequence is very dependent on the feeder setup and component retrieval sequence, and vice versa (Bard et al., 1994). Indeed, the concurrent movement of many machine parts (such as turret rotation, feeder carrier and PCB table) requires a full examination of all feasible combinations of feeder setup and component retrieval sequence in order to determine the best feeder setup and component retrieval sequence for each feasible solution of the component pick-and-place sequence. Moreover, the component pick-and-place sequencing problem is also tightly intertwined with the nozzle optimisation problem where seeking a good component pick-and-place sequence without considering nozzle optimisation might lead to many unnecessary nozzle changes, which is very inefficient. In addition, there are many other issues that should be considered in optimising these sub problems such as the grouping of components in a sub tour (i.e. what are the components that should be picked-and-placed together in each route if there is more than one pipette/nozzle per head); the speed difference among PCB table, feeder carrier and head movement; component transportation time; simultaneous pickup; etc.

The complexity of concurrent machine operations also causes difficulties in formulating a realistic mathematical programming problem (De Souza and Lijun, 1995). Many technical constraints have to be considered. These include:

a) The head, feeder carrier and PCB table usually move independently and at different speeds. Indeed, the speed changes when different sizes of components are to be placed.

b) Smaller size components are usually placed before larger sized components since the larger sized components that have been placed on the PCB may be displaced when the heads and the PCB increase speed in order to place a smaller sized component.

c) Since the head, feeder carrier and PCB table moves concurrently, the three movement delays should be considered simultaneously in order to improve the machine throughput.

Due to the problem size, the mathematical programming approaches are unrealistic. Alternatively, the problem has to be generalised or simplified (Moyer and Gupta, 1996a). For example, Ahmadi (1993), Ball and Magazine (1988), Bard et al. (1994), Chiu et al. (1991), Crama et al. (1996, 1997), Gavish and Seidmann (1987), Leipälä and Nevalainen (1989), Van Laarhoven and Zijm (1993) etc. have abstracted the problem by isolating it into sub problems. Consequently, a heuristic approach, which finds a near-optimal solution in an acceptable time, is more appropriate in solving the problem (De Souza and Lijun, 1995).

2.4 An Overview of Heuristics, Meta-heuristics and Hyper-heuristics

By allowing enough time, an exact algorithm can produce an optimal solution for a combinatorial optimisation problem. However, these algorithms are usually inefficient due to the time they require. In practice, a heuristic solution is highly desirable (Aarts and Lenstra, 1997, 2003; Su and Fu, 1998). Heuristic algorithms can generate good solutions efficiently at a reasonable computational cost but without being able to guarantee either feasibility or optimality (Reeves and Beasley, 1995; Wang et al., 1999). Many heuristics are problem-specific (Reeves and Beasley, 1995). That is, a method will work for one problem domain but is not applicable to solve other problem domains or even other instances of the same problem. More advanced heuristic approaches are called meta-heuristics, which guide local search heuristics to escape from local optima (Reeves and Beasley, 1995; Voss et al., 1999). Some of the common meta-heuristics are genetic algorithms (GA), tabu search (TS), simulated annealing

(SA) and variable neighbourhood search (VNS). The term meta-heuristic refers to a certain class of heuristic methods. Fred Glover first used it and he defines it (Glover, 1997) as follows,

"A meta-heuristic refers to a master strategy that guides and modifies other heuristics to produce solutions beyond those that are normally generated in a quest for local optimality. The heuristics guided by such a meta-strategy may be high level procedures or may embody nothing more than a description of available moves for transforming one solution into another, together with an associated evaluation rule."

Many surveys of heuristic/meta-heuristic approaches such as Foulds (1983), Silver et al. (1980) and Zanakis et al. (1989) attempt to classify the heuristics/meta-heuristics into several broad categories: greedy construction (to generate initial solution) methods, neighbourhood search (improvement) techniques, relaxation techniques, etc. (Reeves and Beasley, 1995). The other common classification of heuristics and meta-heuristics are single solution approaches and population-based approaches (Blum and Roli, 2001). Basic local search (deterministic iterative improvement), simulated annealing, tabu search, variable neighbourhood search etc. are examples of single solution approaches. Whereas genetic algorithms, ant colony algorithms, memetic 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 neighbour, multiple fragment and insertion heuristics (Johnson, 1990). These approaches are often simple but practical as an initialisation 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.

A neighbourhood search, which is usually known as a local search, attempts to improve the solution by exploring the neighbourhood of the present solution (Aarts and Lenstra, 1997). The neighbourhood of a solution is the set of solutions that are close to the current solution in some sense. The local optima are the best solution in each neighbourhood whilst the global optimum is the best solution with respect to the whole solution space. Plateaus are regions where no neighbourhood is better than any other. The important decision in local search is of deciding the neighbourhood structure(s) and how to explore solutions in the neighbourhood. Basic local 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. A move in a local search is the change defined by the neighbourhood structure that is made to the current solution in order to produce a neighbour solution. Several local search meta-heuristics are based upon a simple approach called hill-climbing (for maximisation problems) or descent method (for minimisation problems). The hill-climbing approach iteratively inspects the neighbourhood and replaces the current solution with a candidate solution that has a better fitness. Two variants of descent methods are steepest descent and random descent. A steepest descent method iteratively searches the whole neighbourhood and chooses the best neighbour, whilst a random descent approach selects neighbouring solutions randomly and accepts the first improved solution quality (Dowsland, 1995). The descent method is a fast algorithm, however, due to rapid convergence in local optima, it is usually hybridised with other methods, which form meta-heuristic approaches such as tabu search and simulated annealing.

Tabu search (Glover, 1989, 1990) makes use of memory structures and incorporates the deterministic improvement algorithm (i.e. descent method) with the possibility of accepting a worse solution in order to escape from local optima (Aarts and Lenstra, 1997, 2003). It is a systematic search approach that exploits adaptive memory structures (Glover et al., 1993; Glover and Laguna, 1997). The best legal neighbour 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 neighbours is restricted by a tabu list. However, an illegal neighbour that attains a certain aspiration level can still be accepted.

Simulated annealing (SA) is motivated from the analogy between combinatorial optimisation problems and the physical annealing of solids (crystals) (Aarts, 1989) 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 pyhsical energy of the solid as the objective function whilst the ground state is a globally optimal solution. The main idea of simulated annealing is to accept all improving solutions whilst 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.

Genetic algorithms (GA) are a population-based method inspired by the principles of natural evolution (Goldberg, 1989; Man et al., 1999). 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 chooses the parents based on survival of the fittest. That is, the better fitness values are more likely to be chosen to undergo reproduction in order to produce offspring.

Variable neighbourhood search (VNS) is a relatively unexplored approach (Hansen and Mladenović, 2001). It was introduced by Hansen and Mladenović (1997). By systematically changing the neighbourhood within a local search algorithm, VNS can explore distant neighbourhoods of the current solution, and jump to a new solution if it is superior to the current solution (Hansen and Mladenović, 1997). A local search is applied repeatedly to obtain the local optima from the selected neighbouring solution. A detailed discussion of VNS approaches is given in chapter 7.

More recently, the term hyper-heuristic has been introduced. Burke et al., (2003a) define the term hyper-heuristic as follows:

"*Hyper-heuristics can be broadly described as the process of using (meta-) heuristics to choose (meta-) heuristics to solve a given problem*."

A hyper-heuristic is not a problem-specific approach (Burke et al., 2003a, 2003b; Cowling et al., 2001a, 2001b, 2002a, 2002b, 2002c) but aims to be a general-purpose heuristic that can handle a wide range of problems. A more detailed discussion of hyper-heuristics is presented in chapter 6.

Heuristic/meta-heuristic techniques such as genetic algorithms, tabu search, simulated annealing and greedy search have demonstrated successful results in solving the PCBA optimisation problem. Khoo and Loh (2000), for example, have developed a prototype genetic algorithm to enhance a planning system for the placement of surface mount devices (SMDs) on a Fuji FCP-IV. Wang et al. (1999) argued that their genetic algorithm performs as well as a human expert in optimising the feeder slot assignment problem for the Fuji QP-122. A knowledge-based system that can reduce human intervention and increase the integration of various decisions within the PCBA system is also suitable for solving the optimisation problems of an SMD placement machine (De Souza and Lijun, 1995). Knowledge-based approaches for solving the optimisation problems posed by SMD placement machines have been proposed by Cavalloro and Cividati (1988) who used expert systems (Jackson, 1999), Chang and Terwilliger (1987) who employed a rule-based approach, De Souza and Lijun (1995), Srinivasan and Sanii (1991) and Yeo et al. (1996) who also employed a rule-based approach. A comprehensive survey of the optimisation of single SMD placement machines can be found in chapter 3.

2.5 Summary

The general printed circuit board assembly (PCBA) problem is at least as complex as the TSP, which is known to be NP-complete. This indicates the necessity of using heuristic/meta-heuristic approaches for solving the component pick-and-place scheduling problem. This chapter has briefly described the common terms used in the PCBA in order to underpin the rest of the thesis. In order to provide a consistent background of meta-heuristics, some of the standard heuristics/meta-heuristics that are commonly used in solving the SMD placement machine optimisation problem have been briefly discussed.

The next chapter surveys a single SMD placement machine optimisation problem and classifies the SMD placement machines into five categories based on the machine characteristics and operational methods.

Chapter 3

Survey of Single Surface Mount Device Placement Machine Optimisation in Printed Circuit Board Assembly

3.1 Introduction

Crama et al. (2002), Jeevan et al. (2002) and Sun et al. (2004) agree that the technological characteristics of the placement machine influences the nature of some of the planning problems to be solved and the formulation of the associated models. As a result, little consensus exists as to what a suitable model should be for the characteristics of a given machine, and the formulations, proposed by different authors tend to be difficult to compare. Hence, this chapter will survey the relations between models, assembly machine technologies and heuristic methods. More specifically, this chapter reviews the optimisation of single surface mount device (SMD) placement machines. The work presented in this chapter has been disseminated as follows:

Ayob, M., Cowling, P. and Kendall, G.(2002a) Optimisation for surface mount placement machines. *Proceeding of the IEEE ICIT'02*, Bangkok, 11-14 Dec. 2002, 498-503.

3.2 Surface Mount Device Placement Machine

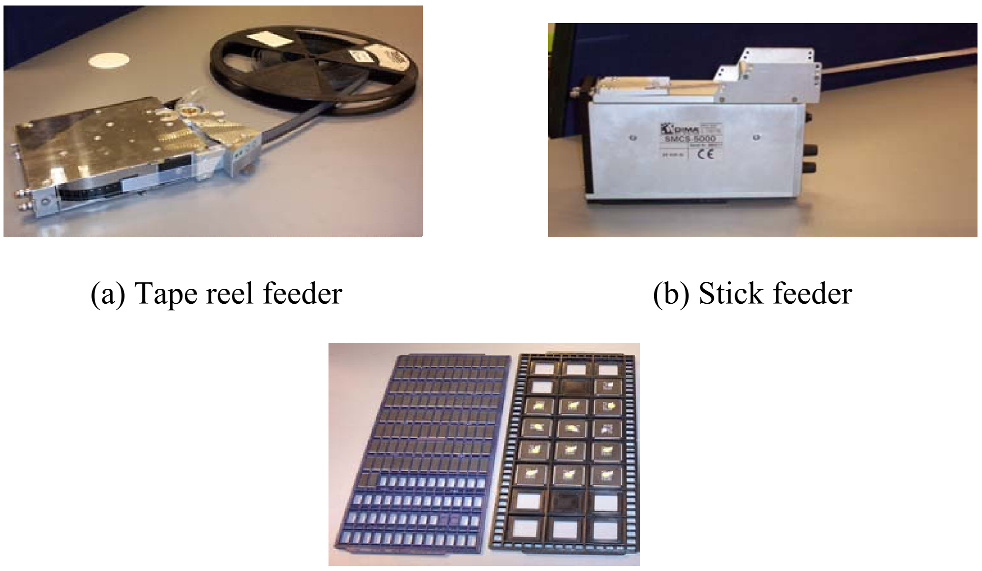
Industrial robotic placement machines have already been classified by mechanical structure such as cartesian/gantry, cylindrical, spherical, single compliance robot for assembling (SCARA), articulated and parallel (Samsung, 2001). In addition, Moyer and Gupta (1996a, 1996b, 1997) also defined three types of typical SMD placement machine; these being SCARA, cartesian/gantry and high speed chip shooter (HSCS). The SCARA is usually known as pick-and-place, which has three joints that permits greater flexibility within the work area. Generally, the SCARA is recommended for high mix, low volume assemblies as well as for odd shape components (Moyer and Gupta, 1998). The cartesian/gantry SMD placement machine has better throughput compared to SCARA. However, Moyer and Gupta (1996a, 1996b, 1997) do not discuss the machine specification and operation. The HSCS placement machine has a turret head that rotates between fixed pickup and fixed placement locations. However, the mechanical structure classifications do not greatly influence the nature of optimisation problems. Moreover, in each category, there are various specification and operational methods of placement machines.

There was an attempt to classify the placement machines based on basic operational methods, these being concurrent and sequential by McGinnis et al. (1992), or fixed pick-and-place point (FPP) and dynamic pick-and-place point (DPP) by Wang et al. (1998). However, these two categories are too broad. Hence, it tends to be difficult to formulate optimisation problems based on these categories. More recently, Magyar et al. (1999) classified the placement machines into three categories, these being insertion, pick-and-place and rotary turret machines; whereas Bentzen (2000) classifications were turret head, pick-and-place and pick-and-place with rotary head. Jeevan et al. (2002) classified them as multi-head, high speed chip shooter machine (HSCS) and robotic arm placement machine. However, they do not explicitly discuss the machine characteristics and the operational methods. Again, the three categories are too broad, which causes difficulty for other researchers when employing the proposed approaches from the literature in order to solve their SMD placement machine problem due to the complexity and concurrent operation of the machine. In addition, each SMD placement machine might have a unique scheduling problem. Thus, this work proposes five categories of machines based on their specifications and operational methods; these being dual-delivery, multi-station, turret-type, multi-head and sequential pick-and-place SMD placement machines. This grouping aims to guide future researchers in this field in order to have a better understanding of the various machine specifications and operational methods, and subsequently are able to use them to apply or even design heuristics, which are more appropriate to the machine characteristics and the operational methods.

Generally, each placement machine has a feeder carrier (or feeder magazine), PCB table, head, nozzle (tool or gripper) and a tool magazine. The feeder carrier, PCB table and head can either be fixed or moveable depending on the specification of the machine. In some cases, the feeder carrier is divided into different feeder banks, each consisting of feeder slots (Wang et al., 1999). Each feeder bank consists of several feeder slots where the component feeders are located. The feeders are used to provide the machine with a continuous supply of components. Several kinds of component feeders are available to handle the various types of component packaging; tape, sticks and trays (or waffle).

A typical feeder carrier consists of either several tape reels or vibratory ski slope feeders or both (Ahmadi et al., 1988; Jeevan et al., 2002). The feeder reels or vibratory ski slope feeders are positioned in the feeder slots according to the arrangement given by the feeder setup. The component feeders might have different widths and several slots may be occupied by a component feeder (Sun et al., 2004). Figure 3.1 shows a few types of component feeders (pictured at the Dima factory).

Tape reel feeders are used to feed components packed in embossed, paper or surf tape. Depending on the component size, the typical tape widths are 8 mm, 12 mm, 16 mm, 24 mm, 44 mm, 56 mm and 72 mm (Bentzen, 2000). If the components are supplied in sticks or tubes, then the stick feeders are used to feed the components. The two common mechanisms of feeding the stick feeders are vibrating and ski-slope. Due to a delicate handling of stick feeders, Bentzen (2000) recommended avoiding using components with stick feeders for mass production. The large size components supplied in trays are fed using tray feeders. Some machines allow a single tray to be placed into the machine feeding area whilst others use an automatic tray-handling unit.



(a) Tape reel feeder (b) Stick feeder

(c) Tray feeder

Figure 3.1: Example of component feeders.

The pipette (or spindle) is located at the head and used to hold a nozzle. It can move in *Z* direction (up-down) to perform pick-and-place operations. The nozzle is used to grasp the component from the feeder and then mount it on the PCB (Altinkemer et al. 2000). Due to the various component packaging, different nozzle sizes are required to handle them and an automatic nozzle change system is used to ensure that the correct nozzle is used. A tool bank is required to provide the exact nozzle size. Usually, vacuum nozzles are used to transport components from component feeders whereas special nozzles with mechanical alignment are required for the handling of odd-shape components (Bentzen, 2000). Figure 3.2 (adopted from Bentzen, 2000) shows different sizes of vacuum nozzles.

The placement arm, that is equipped with head(s), is responsible for picking and placing components. Each head may have more than one nozzle and each machine may have more than one head. There are various types of placement heads, such as a rotating turret head, or a positioning arm head (Wang et al., 1999). The PCB table is required to position the PCB(s) during the placement operation. The table could be stationary, a conveyor system, or an X-Y motion table.

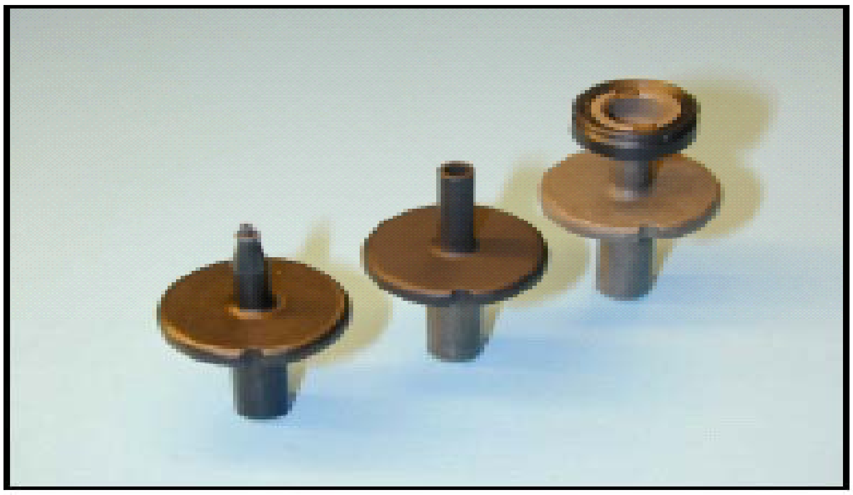


Figure 3.2: Example of vacuum nozzles.

3.3 Surface Mount Device Placement Machine Classification

Based on the specification and operational methods, we have classified the SMD placement machines into five categories. These being: dual-delivery, multi-station, turret-type, multi-head and sequential pick-and-place.

3.3.1 Dual-Delivery Placement Machine

Typically, this machine consists of the PCB table, which can move in both X and Y directions and should be aligned under the head to perform the placement operation; the placement arms and two component delivery carriers are only able to move in the X-direction (Ahmadi et al., 1988,1995). The pick-and-place heads are mounted at the two ends of a fixed length arm, which can move between two fixed positions in the Y-direction only. The unique and important feature of this machine type is that each pick-and-place operation alternates between two sides, i.e. while one head is performing the pick operations, the other one is placing components on the board (Ahmadi et al., 1995; Safai 1996). For this machine, all movements of the PCB table and feeder carrier are frozen during the pick-and-place operations. Therefore, the maximum time taken by the arm, PCB table and feeder carrier movements will determine the cycle time.

The machine used by Tirpak et al. (2000), the Fuji NP-132 (see figure 3.3), is another variant of a dual-delivery SMD placement machine. It has dual turret placement heads mounted on the two overhead servo-driven X-Y gantries. Each head is equipped with an internal camera for on-the-fly vision inspection and 16 nozzles. The pick-and-place operation can begin once the PCB has been loaded into one of the conveyers and the fiducial marks check has been performed. First, the gantry moves to position the turret head for the first component pickup (assuming the head is equipped with the correct nozzles, otherwise nozzle changes are required). Next, the turret head rotates to locate the appropriate nozzle. Then the component is picked up from the feeder. This process is repeated until the turret head has rotated by 360 degrees and all nozzles are holding components (or left empty due to incompatibility with the components etc.). Next, the gantry moves and locates the head to place the first component and meanwhile the turret also rotates to position the appropriate component at the correct placement point. These steps are repeated for the next locations on the board that have to be placed on the same tour. While the first head is placing components, another can concurrently pick components. To avoid collision, only one head can perform placement operations at a time.

The Dynapert MPS 500 (Ahmadi et al., 1988, 1991 and 1995), the Panaset MCF that is equipped with 10-nozzle gang pickup (Panasonic, 2001), the Fuji NP-132, which contains dual turret placement heads with 16 nozzles on each head (Tirpak et al., 2000), the Fuji IP2 that has 2-nozzle on each head (Safai, 1996) and the Siemens Siplace 80S-20 (Tirpak et al., 2000) are all examples of dual-delivery placement machines.

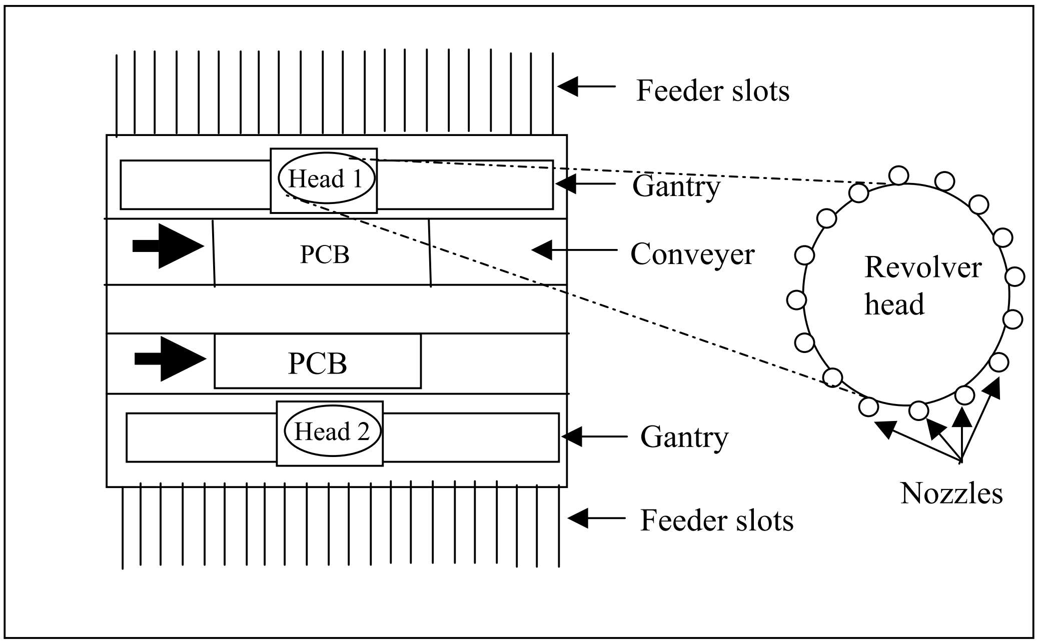


Figure 3.3: A dual-delivery SMD placement machine.

3.3.2 Multi-Station Placement Machine

In general, this machine has more than one placement module (or stations) each one being mechanically identical and able to assemble electronic parts concurrently. The stations are connected by a conveyor system to transfer boards among stations. The PCB is fixed to the pallet and then transferred through the stations by a pallet circulating system ("conveyor")(Csaszar et al., 2000a). Each station receives all the necessary pick-and-place coordinate data for one machine cycle (the interval between two conveyor steps), and completes the cycle's placement sequence autonomously and concurrently with the other stations. After all stations have finished, the conveyor is moved, and the placement procedure continues. The Fuji QP-122 (Wang et al., 1999; Csaszar et al., 2000b) that has 16 stations with each station consisting of fixed multi-feeder unit and a single-nozzle placement head is an example of the multi-station placement machines. Figure 3.4 shows a sketch diagram of a multi-station SMD placement machine (adopted from Csaszar et al., 2000a, 2000b).

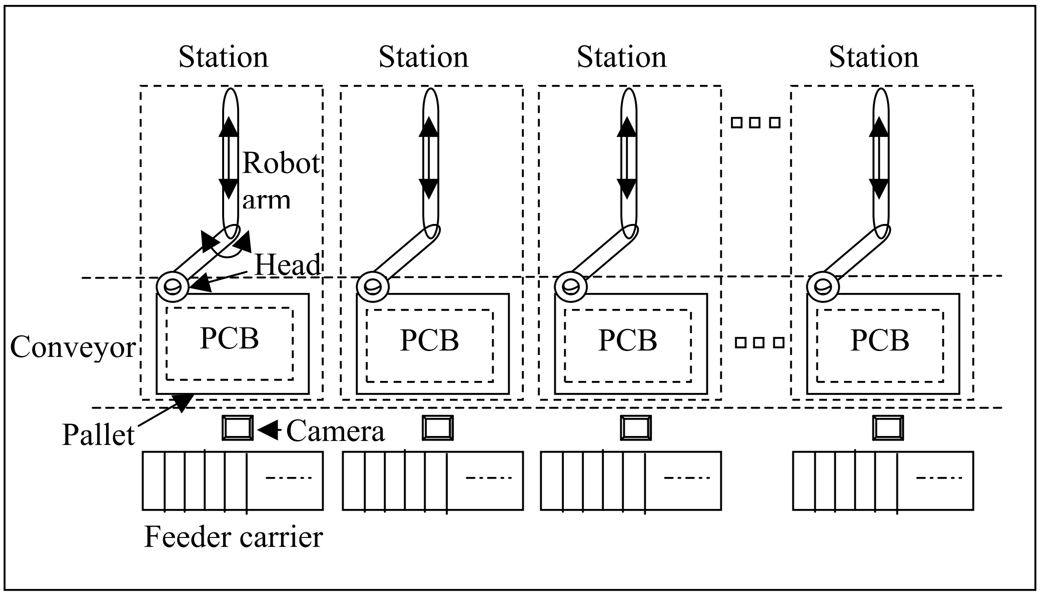


Figure 3.4: A multi-station SMD placement machine.

3.3.3 Turret-type Placement Machine

This machine is usually called a chip shooter machine (Ho and Ji, 2003; Moyer and Gupta, 1996a, 1996b, 1997, 1998). The machine uses a placement mechanism mounted on a rotating turret (drum or carousel), with multiple placement heads, that rotate between a fixed pickup and fixed placement locations (Burke et al., 1999; Bentzen, 2000; Gastel, 2002).

Generally, each pick-and-place operation starts by retrieving a component at the grip station, while the placement station simultaneously mounts (if there is a component to be mounted) a component at a pre-specified location on the PCB (Ellis et al., 2001; Klomp et al., 2000). Then, the feeder rack moves to get the next appropriate feeder in position, and the PCB table simultaneously moves to position the next location under the place station. In fact, the movements of the PCB table, feeder carrier and turret may take place concurrently (Crama et al., 1996). Actually, the i^{th} placement operation and the $(i+k)^{th}$ pickup operation (where k is the half of the sum of available heads) do not have to be performed concurrently, but are required to be done between the same two turret rotations (Crama et al., 1996). Typically, this machine has 12 to 24 placement heads, each equipped with three to six nozzles, which can be changed on-the-fly (Gastel, 2002). Due to the modus operandi of the machine, the rotating turret is only capable of simultaneously holding up to half of the sum of available heads.

As the turret rotates, several parallel operations are performed. Before arriving at the placement station, the picked component will undergo the following operations; a visual inspection of the component for orientation and diagnostics; a component's ejection if the picked component is rejected, or otherwise the component is oriented for placement (Bard et al., 1994). After passing the placement station, the nozzles are set up and reoriented for the next pickup operation. These are parallel operations that are dominated by the pickup-and-placement operations (Bard et al., 1994). In general, the PCB table can immediately move to position the next placement point at the placement station once the current placement operation has been completed. Similarly, the feeder rack can immediately move to position the next component at the pickup station after the completion of the current pickup operation. However, the turret rotation can only start after both the pickup-and-placement operations have been completed. In practice, the PCB table movement is the determining factor (in most cases) of the throughput rate of turret-type SMD placement machines compared to the turret rotation time (Gastel, 2002).

Some of the common turret-type placement machines include the Fuji FCP-IV (Kumar and Luo, 2003; Crama et al., 1997) that has 12-nozzles mounted on a rotary head, the Fuji CP4, CP4-2, and CP4-3, which have 12 placement heads, the Fuji CP6, which has 20 placement heads (Fuji, 2001), the CM82 is equipped with 18 placement heads (Ohno et al., 1999), the Fuji CP II has 12 placement heads where each head is equipped with 2 nozzles of different sizes (however, only one is used at a time depending on the specification of the component to be picked up) (Bard et al., 1994) and Panaset MKI-LL (Grotzinger and Sciomachen, 1988).

Gastel (2002) had addressed some disadvantages of the turret-type placement machine:

1) The movement of the PCB table is imposed by the acceleration forced on the pre-mounted components. If larger size components (referred to as a slow component) have been placed onto the PCB, then movement of the PCB table will become slower.

2) The accuracy of the machine is limited by the movement of the PCB table and the vibration from the moving feeder carrier.

3) The use of a tray feeder is not possible.

4) An intelligent motorised feeder is required to perform pick corrections for small components.

5) Due to the moving feeder carrier, a long footprint is required by the machine.

Due to a restriction of the mechanical structure of the turret head, the machine is also not capable of performing a simultaneous pickup or simultaneous placement.

Figure 3.5 shows a sketch diagram of the Fuji CP IV/3, a turret-type SMD placement machine (adopted from Klomp et al., 2000).

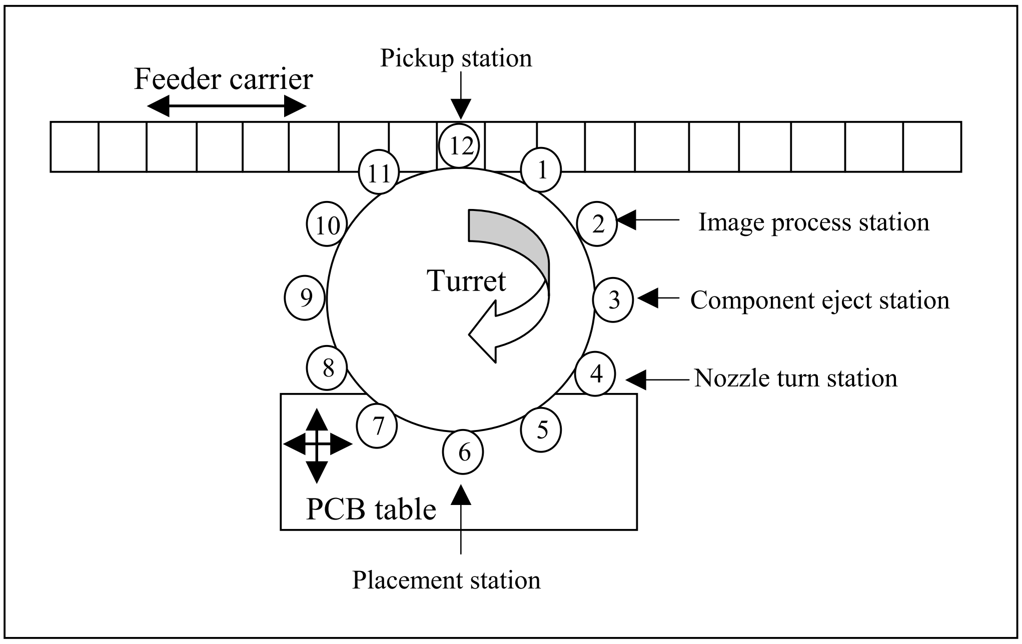


Figure 3.5: A sketch diagram of the Fuji CP IV/3 (a turret-type SMD placement machine).

3.3.4 Multi-Head Placement Machine

Bentzen (2000) refers to the multi-head placement machine as a pick-and-place machine. The multi-head placement machine is the most flexible machine that can handle a wide range of component packages (Bentzen, 2000; Van Laarhoven and Zijm, 1993). The multi-head placement machine differs from the turret-type in the component transportation mechanism (Bentzen, 2000). It uses an X-Y gantry head to transport components from fixed feeders and then place them onto the fixed PCB whereas the placement head of a turret-type machine is rotated to pick up the component at the fixed pickup location from a moveable feeder carrier and placing it onto the fixed placement location of moveable PCB. Figure 3.6 shows an example of a multi-head placement machine.

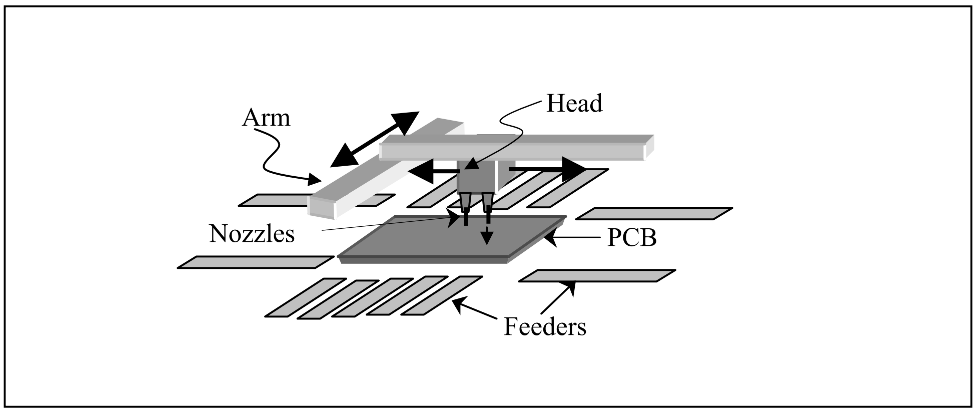


Figure 3.6: A multi-head SMD placement machine.

There are two types of multi-head placement machines (Jeevan et al., 2002). The first type has a stationary PCB table and feeder carrier with the arm and head being able to move concurrently in the X (horizontal) and Y (vertical) directions, respectively, to perform the pick-and-place operations. Another type has an X-Y motion table and moveable feeder carrier with the arm and head travelling between the fixed pickup-and-placement locations (Jeevan et al., 2002).

The tour of the heads begins by picking up a few components from the feeder (assuming the heads are equipped with the correct nozzles, otherwise nozzle changes are required) simultaneously or sequentially (depending on the pickup positions). Then, the head and the arm travel (in the X and Y direction simultaneously for the first type of multi-head placement machine) to position itself on top of the point where the component will be placed, and then the head moves down (Z-direction) and places the component on the board before returning to the original position and repeating these steps for the next locations on the board that have to be placed on the same tour. After completing a tour, the head returns to the feeder location to begin another tour, unless nozzle changes are required. The heads of this machine can be similar to the heads of turret-type machine. The difference is that it is located on top of the arm (Altinkemer et al., 2000) and the pickup-and-placement locations are not necessarily fixed.

The Quad 400 series (Altinkemer et al., 2000) is an example of multi-head SMD placement machine.

3.3.5 Sequential Pick-and-Place Machine

According to Kumar and Li (1995), a typical machine of this type has a placement head mounted at the end of an arm. The arm can move in the X-direction, whilst the head can move simultaneously in the Y-direction. The pipette/nozzle on the head can move in the Z-direction to perform pick-and-place operations. The placement arm starts by moving to the tool magazine to equip itself with the proper nozzle. Next, it moves to pick a particular component from the feeder location, and then place the component at the appropriate location on the board. If the following component uses the same nozzle type, the arm moves directly to the feeder slot to perform the subsequent pick-and-place operation. Otherwise, the arm goes to the tool magazine for automatic nozzle changes (Kumar and Li, 1995; Loh et al., 2001). The Quad IIIC is an example of this machine type (Loh et al., 2001). Figure 3.7 shows a sketch of a sequential pick-and-place SMD placement machine. Previous works, which focused on this machine type, do not mention the feeder carrier and PCB table movements. Therefore, we assume both of them can be stationary or movable.

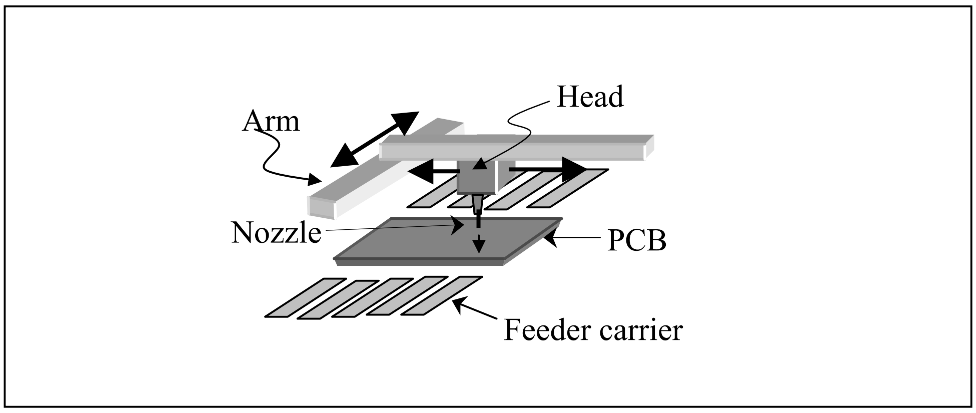


Figure 3.7: A sequential pick-and-place SMD placement machine.

3.4 Production Planning Problem in Printed Circuit Board Assembly

Crama et al. (2002) provided an exhaustive survey of some of the major optimisation problems arising in the area of production planning for the assembly of PCBs. By considering a long-term decision with the *mixed demand* and the *fixed shop layout,* Crama et al. (2002) classified the production planning problems into eight sub problems; these being (1) assigning PCB types to product families and to machine groups, (2) allocating component feeders to machines, (3) partitioning component locations on the PCB to indicate which components are going to be placed by each machine (for each PCB type), (4) sequencing the PCB types, (5) assigning component feeders to slots on the feeder carrier (feeder setup), (6) sequencing the component pick-and-place operations, (7) component retrieving plans and (8) a motion control specification. Normally, the decision as to which of these sub problems is to be solved is based on which sub problem will minimise the assembly cycle time (Crama et al., 2002). However, the dilemma is that all the sub problems are intertwined and the question arises as to which one should be solved first. As a consequence, some researchers tackled the problem in an iterative manner, instead of a one-pass procedure through each of the sub problems. The technological characteristics of the SMD placement machine can also influence the nature of some of the problems to be solved and the formulation of the associated models (Crama et al., 2002; Moyer and Gupta, 1997). Crama et al.

(2002) also addressed the problem of having insufficient problem descriptions in the literature by suggesting that all authors should mention (at least) the following topological elements in their papers:

1) Shop layout (decoupled workcells, one or several assembly lines, etc.);
2) Characteristics of the product mix (high volume-low variety, low volume-high variety, etc.);
3) Setup policy if more than one board type is to be produced;
4) Relevant characteristics of the SMD placement machines (sequential, concurrent, etc.);
5) Decision to be taken, according to the eight sub problems.

Other excellent surveys have been conducted by Ahmadi (1993), Ji and Wan (2001) and McGinnis et al. (1992). Ahmadi (1993) devised a hierarchy of decision problems and developed the model to optimise the decision making process in PCB manufacturing. Ji and Wan (2001) and McGinnis et al. (1992) categorised the production planning problems into three stages; grouping (i.e. assigning PCB types to product families and to machine groups); allocation (i.e. identifying which machine in the assembly line to assemble which components); and arrangement and sequencing (i.e. assigning component feeders to slots on the feeder carrier and sequencing the component's pick-and-place operations).

To complement these surveys, this chapter also extensively reviews a single machine optimisation problem that highlights some major optimisation issues in each sub problem.

3.5 Single Machine Optimisation

In this section, we focus on the problem of a single machine with a single board type by assuming that the other sub problems have been solved. In the context of a hierarchical decomposition approach, the single machine optimisation problem is considered as the lowest operational level (Magyar et al., 1999). To date, the single machine optimisation problems still cannot be solved efficiently by the PCB machine vendors and software companies (Magyar et al., 1999).

Crama et al. (2002) classified the single machine problem into four sub problems; these being feeder setup, component pick-and-place sequencing, component retrieval plan and motion control. Whereas, in this work, we add one more sub problem that is a nozzle optimisation (i.e. five sub problems in total). On the other hand, Magyar et al. (1999) encountered four sub problems of the single machine optimisation. These being: feeder setup, component pick-and-place sequencing, component retrieval and nozzle optimisation. Indeed, these sub problems are also tightly intertwined. As a result, some researchers solved the problem in iterative manner, instead of a one-pass procedure through each of the sub problems and some used an integrated approach. Some works have addressed the problems of feeder setup and pick-and-place sequencing independently by making assumptions about the rest of the problem, and some prefer to solve both problems as an integrated solution (Ellis et al., 2001). A hierarchical problem solving approach has also been studied (Magyar et al., 1999). Nevertheless, many researchers also tackled the sub problems independently (assuming that the other sub problems have been solved).

3.5.1 Motion Control

When considering an SMD placement machine that has a moveable head, a feeder carrier and a PCB table, one should consider where are the effective pick-and-place points. That is, where the robot arm meets the feeder carrier (or the PCB) to pick (or place) components. The robot (that is the arm and head) is able to move in both X and Y directions concurrently to pick-and-place a component. The feeder carrier and the PCB table are moveable in the X-axis to position the component pickup coordinate and the placement coordinate of the PCB, respectively. The robot, PCB table and feeder carrier can move concurrently. The robot travels between feeder carrier and PCB table for picking and placing a component, respectively.

Up until now, there have not been many research works reporting on improving the motion control. This might be because this decision is directly relevant to the production preparation (Van Laarhoven and Zijm, 1993). In fact, many SMD placement machines use fixed pick-and-place points since not many of them have moveable heads (X-Y), feeder carriers and PCB tables. For example, a turret-type SMD placement machine has a rotating turret, which rotates from a fixed pickup location to the fixed placement location.

Some works that have focused on motion control are Bonert et al. (2000), Fu and Su (2000), Hop and Tabucanon (2001a, 2001b), Su et al. (1995), Wang (1996) and Wang et al. (1997, 1998). These works suggest a dynamic pick-and-place (DPP) point to avoid robot idling time. The approach allows the robot to pick-and-place a component at any location rather than a fixed pickup-and-placement (FPP) location. Most of these works solved the problem for sequential pick-and-place machine except Bonert et al. (2000), which dealt with a dual-delivery placement machine. Details of these works are discussed in chapter 4 (see section 4.2).

3.5.2 Nozzle Optimisation

Nozzle optimisation, in the context of single machine optimisation, involves searching for an effective nozzle (tool) assignment and sequencing/switching. When the SMD placement machine has more than one nozzle per head (or even a single nozzle per head), choosing an effective nozzle group (or a nozzle) is important in order to improve the pick-and-place operations and to minimise the number of nozzle change operations. Having a proper nozzle group assignment might lead to having more simultaneous pickup operations, minimise feeder carrier movement, as well as robot arm and/or PCB table movements. This can ultimately improve the machine throughput. A nozzle changeover operation is very time consuming (Crama et al., 1990; Jeevan et al., 2002; Lee et al., 1999; Magyar et al., 1999; Safai, 1996; Shih et al., 1996). Optimising the pick-and-place operation without considering the nozzle switching operations may not be efficient since it may cause many unnecessary nozzle changes that will significantly reduce machine throughput (Magyar et al., 1999). The problem of minimising nozzle switching and minimising the pick-and-place operations are tightly intertwined and should not be solved independently. Our industrial partner (Dima SMT Systems) also agreed that these problems should not be solved separately. Nevertheless, the nozzle changeover operation is not directly affected by the component allocation and feeder setup decision (Sun et al., 2004).

In minimising the dual-delivery placement machine, Tirpak et al. (2000) defined a nozzle minimisation problem as an assignment of a nozzle based on the weights associated with each nozzle type. This involves finding the best distribution of the nozzle on the heads, which yields the minimum pickup time.

To date, there has been relatively little research that has addressed the minimisation of nozzle switching. Even an exhaustive survey by Crama et al. (2002) did not address this problem. A survey by McGinnis et al. (1992) also found that only a small amount of research employ component specific nozzles. For example, Bard (1988), Chandra et al. (1993), Crama et al. (1990, 1994), Shakeri (2004), Sule (1993) and Tang and Denardo (1988a, 1988b) all considered minimising tool switches in the context of flexible manufacturing. Crama et al. (1990) proposed a heuristic hierarchical approach to the problem of minimising the throughput rate of a line of several SMD placement machines by first assigning the nozzle to the machines, and then performing the component allocations. Again, this is a tool management issue in the context of flexible manufacturing rather than a single machine minimisation problem. A crucial problem of tool management (minimising) in flexible manufacturing is to identify the sequence of parts to be produced, and what tools to allocate to the machine so as to minimise the number of tool setups (Crama et al., 1994).

As far as we are concerned, none of the research (in the context of a single SMD placement machine minimising) has tackled the nozzle minimisation problem individually. Some works that addressed the importance of nozzle minimising are Ahmadi et al. (1988, 1991), Chang and Terwilliger (1987), Crama et al. (1990), Jeevan et al. (2002), Magyar et al. (1999), Safai (1996), Shih et al., (1996) and Tirpak et al. (2000). They solved the nozzle minimisation problem together with the problem of sequencing the pick-and-place operation and/or feeder setup.

Chang and Terwilliger (1987) proposed a rule-based approach to solve the component placement sequence problem. One of the rules aims to minimise the nozzle changes. Unfortunately, they did not present any results.

An expert system approach has been developed by Shih et al. (1996) to minimise a multi-station SMD placement machine by first emphasising the minimisation of nozzle changes, then minimising the component pick-and-place sequence. The nozzle minimisation procedure minimises nozzle changes by grouping the components in a placement sequence so as the components using the same nozzle type can be placed consecutively. They employed five rule sets for nozzle minimisation, these being:

1) Rule set 1: Grouping the placement steps based on the station where the placement steps will be performed. Next, the nozzle minimisation procedure can begin with respect to the individual station.
2) Rule set 2: Sequencing the nozzle sets used based on their handling capabilities such that more clearance is provided in placing large components while simultaneously minimising the frequency of nozzle changeovers. The size of the nozzle affects the clearance required at a placement location on the PCB.
3) Rule set 3: Arranging the nozzle changeover in ascending order of component mass.
4) Rule set 4: Ensuring the placing of *unleaded components* (components without legs) prior to leaded components.
5) Rule set 5: Sequencing the component placement in ascending order of component mass.

Based on the output from the nozzle minimisation stage, Shih et al. (1996) employed a simple descent search algorithm to minimise the component pick-and-place operation. Their results was verified by machine experts and showed an improvement of 5.72% in terms of component placement time, which might contribute to about 15 working days of time saving over a year.

Safai (1996) does not explicitly consider how to minimise nozzle change operations. In fact, Safai (1996) indirectly reduced the nozzle changes when eliminating the head contention (i.e. the case when more than one head required the same nozzle at the same time). However, in order to minimise the nozzle changes, so as to minimise the assembly cycle time, Safai (1996) represents the cost of nozzle changes in terms of placement cost and includes the cost in the objective function.

Magyar et al. (1999) created the nozzle usage table to identify the nozzle layers. They considered the trade-off between minimising the nozzle changes and minimising the number of placement groups (i.e. sub tours). Generally, reducing the nozzle changes will increase the number of placement groups, and vice-versa. The nozzle changes are costly, likewise additional sub tours increases the camera costs. Their algorithm iteratively creates good nozzle layers by increasing (which started with minimum nozzle changes) the number of nozzle changes and determining the number of sub tours.

By listing the type of components to be assembled and the associated nozzles used, Tirpak et al. (2000) assigned the nozzles to the heads by considering the best distribution of the nozzles to the head. They improved the initial nozzle setups by randomly selecting two nozzles of different sizes, and swapping their positions on the revolver head (each revolver head has 16 nozzles).

Recently, Jeevan et al. (2002) used a genetic algorithm to minimise the component pick-and-place sequence of the multi-head SMD placement machine by considering the importance of minimising the tool change operation. They represent a distance of a TSP tour (i.e. a total pickup-and-placement distance) as a fitness function. In order to eliminate any unnecessary nozzle changes, they use all components that can be placed by a certain nozzle before changing the appropriate nozzle. However, since a component type (or package) can be picked up by more than one nozzle type, and the tool changing time was excluded from the fitness function evaluation, the aim of reducing tool changes operation might not be fulfilled. Indeed, selecting a good nozzle sequence is important for reducing nozzle change operations in order to enhance the SMD placement machine throughput.

3.5.3 Component Pick-and-Place Sequence Optimisation

Suppose that the feeder setups, the component retrieval plan, the motion control and the nozzle sequencing have been determined. In this case, we need to search for a good component pick-and-place sequence in order to maximise the machine throughput. Many papers (for example, Jeevan et al., 2002 and Leu et al, 1993) defined the component pick-and-place optimisation as finding a shortest route to pick-and-place the electronic components onto the PCB. This is only true if other factors such as nozzle changes, feeder transportation time (i.e. time taken by the feeder to transport the component to the pickup position), gang pickups (i.e. simultaneous pickup) etc. are ignored. Therefore, it is more precise to define the component pick-and-place optimisation as finding a shortest time to pick-and-place the electronic components onto the PCB (Ng, 1998).

Due to the advancement of current high-tech products, the component density on the PCB is increased. That is the distance among the PCB points tends to be smaller. As a result, the decision of feeder setups and pickup sequences are more crucial in determining the efficiency of the machine compared to the component placement sequencing (Sun et al., 2004). However, we also found that minimising the nozzle change operations is a crucial decision too. Based on our discussion with PCB assembly companies, it is a common practice not to frequently change the feeder setup unless it is unavoidable. Therefore, in this case, the optimisation of the component pickup-and-placement sequencing plays a significant factor in improving the throughput of SMD placement machines.

Generally, many researchers modelled the component pick-and-place sequencing problem as a travelling salesman problem (TSP) (Bard et al., 1994; Chan and Mercier, 1989; Chan, 1993; De Souza and Lijun, 1994; Drezner and Nof, 1984; Duman and Or, 2004; Foulds and Hamacher, 1993; Francis et al., 1994; Gavish and Seidmann, 1988; Jeevan et al., 2002; Khoo and Ng, 1998; Kumar and Luo, 2003; Leipälä and Nevalainen, 1989; McGinnis et al., 1992; Tirpak et al., 2000) whilst Ball and Magazine (1988) treated it as a rural postman problem. Nevertheless, many researchers solved the pick-and-place sequencing problem as a unique problem since the problem relies heavily on the machine characteristics (Ho and Ji, 2003). The optimisation of a TSP aims to find a route of visiting each city exactly once while minimising the total distance travelled (Keuthen, 2003). By defining chip locations (PCB points) as cities and the time between chip insertions (or placements) as distances, the component pickup-and-placement sequencing problem can be formulated as a TSP (Chan and Mercier, 1989). However, in reality, it is not a straight forward mapping since the effectiveness of the component pick-and-place sequence is not only dependent on the robot (and/or PCB table, feeder carrier) travelling distance. As discussed in section 3.4.2, the nozzle optimisation problem is tightly intertwined with the problem of sequencing the pick-and-place operation. Without considering nozzle switching, optimising the pick-and-place sequencing operation might cause many unnecessary nozzle changes, which will not produce an effective schedule. A rural postman problem, which is a generalisation of the Chinese postman problem, is a problem of finding an optimum (or least-cost) postman tour covering all the edges (streets) in the network, in which the underlying street network may not form a connected graph (Pearn and Wu, 1995; Kang and Han, 1998).

Ball and Magazine (1988) was the first work, which attempted to solve the component pick-and-place problem for a moving head, stationary PCB table and feeder carrier. Thereafter, many studies optimising the component pick-and-place problem have been reported.

Leu et al. (1993) associated the planning problem (particularly the component pick-and-place sequencing problem) with the characteristics of various SMD placement machines. They pointed out three planning problems as shown in table 3.1. The third column in table 3.1 shows the relation between the machine characteristics (Leu et al., 1993) and the classification of the SMD placement machine as in section 3.2. However, due to a technology change, the SMD placement machine characterised for TSP is not classified in section 3.2. This might be an old machine and as far as we concerned, none of the work focuses on this machine type except Leu et al. (1993). The planning problem of the first machine (refer to table 3.1) was treated as a TSP since the issue is to find a sequence of placement head(s) in visiting all PCB point locations such that the travelling time is minimised, regardless of the pickup operations. Whereas, due to the fact that the cost of component placement sequencing is very dependent on the feeder setup, the planning problem of the second machine (refer to table 3.1) was modelled as a Pick-and-Place Problem. Subsequently, the planning problem of the third machine (refer to table 3.1) was modelled as a Moving Board with Time Delay Problem because there is often a time delay caused by either the feeder carrier movement or the turret rotation when the PCB travels between two consecutive placement points and therefore the problem cannot be treated as a simple TSP.

A rule-based approach has been employed by Mettalla and Egeblu (1989) for solving the component pick-and-place problem. The rules were based the on dominance properties of robot arm movement, feeder carrier movement etc.

By considering a case where certain placement sequences are not acceptable (which cause a placement head damaging the previously placed components during a placement operation), Duman and Or (2004) treated the component pick-and-place sequencing problem as a Precedence Constrained Travelling Salesman Problem (PCTSP).

TABLE 3.1: A RELATION AMONG SMD PLACEMENT MACHINE CHARACTERISTICS, PLANNING PROBLEM TYPES AND MACHINE CLASSIFICATION.

Problem type		**SMD placement machine characteristics**	**Machine type**
1	Travelling Salesman Problem (TSP).	Stationary head, X-Y table, direct feeding of components to assembly head.	Unclassified.
2	Pick-and-Place Problem.	Moving head, stationary table, stationary feeders.	Sequential pick-and-Place.
3	Moving Board with Time Delay Problem.	X-Y table, moving feeders, supply of components with a multi-head turret or a moving head between two fixed locations.	Turret-type.

Sanchez and Priest (1991) addressed four basic insertion/placement rules, these being:

1) To avoid interference, smaller size components should be placed prior to larger size components.

2) All the same type of components, are assembled in one pass.

3) Assemble components of identical sizes and shapes, and then assemble the other components of non-similar size and shape.
4) Choose a near-optimal sequence to minimise the PCB table movement.

In other work, Wong and Leu (1993) also addressed four basic component pick-and-place and feeder setup rules that are usually adopted:

1) Sequence the component placement for minimum routing time.
2) Arrange feeder reels so as to minimise component pickup time.
3) Place identical SMDs in one pass.
4) Sequence placement according to component size.

However based on our discussion with an expert from DIMA, one should avoid consecutive pick ups of the same component type from the same feeder slot since this incurs an extra cost in the form of a component feeder transportation cost, which is usually more than the cost of moving to pickup from the other feeder slot. Therefore we would suggest that the pick-and-place sequence should be arranged such that the components that used the same nozzle type(s) are assembled consecutively instead of assembling the same type of components or identical SMDs in one pass. This strategy may help reduce nozzle change operations, which is very time consuming.

Crama et al. (1996) found that a simple forward dynamic programming scheme is not capable of producing an efficient schedule for component placement sequencing. Many other heuristics such as tabu search (Csaszar, 2000a; Su et al., 1998), genetic algorithms (Leu et al., 1993; Khoo and Ng, 1998; Khoo and Ong, 1998), expert systems (Huang and Srihari, 1993), knowledge-based systems (De Souza and Lijun, 1994), rule-based systems (Mettalla and Egeblu, 1989) and neural networks (Su and Srihari, 1996) are among the most effective approaches for optimising the component pick-and-place sequence and feeder setup.

3.5.4 Feeder Setup Optimisation

The question of where (i.e. in which slots) the feeder reels should be placed in each placement machine is referred to as feeder setup (Tirpak et al., 2000), the feeder rack assignment problem (Klomp et al., 2000), component-feeder arrangement (Khoo and Loh, 2000), the reel positioning problem (Ahmadi et al., 1995; Ohno et al., 1999), feeder assignment (Loh et al., 2001; Hop and Tabucannon, 2001b), feeder allocation (Altinkemer et al., 2000), or magazine assignment (Ahmadi et al., 1988). In this work we use the term *feeder setup* to refer to this problem.

The feeder setup decision determines where the component feeders are located on the feeder slots of the feeder carrier/feeder bank. Unlike the travelling salesman problem, the evaluation of the solution quality of the feeder setup is not straight forward (Sun et al., 2004). For example, the 'cost' of a particular feeder setup depends on the sequence of pick-and-place operations (Ball and Magazine, 1988). That is, we need other interrelated decisions such as a decision for nozzle assignment and sequencing, pickup-and-placement sequencing, gantry scheduling etc. Nevertheless, the other decisions should be solved in order to avoid disturbances and maintain the evaluation's consistency while searching for an improved feeder setup (Sun et al., 2004). Similarly, when optimising component placement sequencing, other optimisation problems such as a feeder setup are solved in a simple manner or by leaving it fixed (i.e. assumes it has already been solved). For example, Moyer and Gupta (1996a), Dikos et al. (1997) and Leipälä and Nevalainen (1989) solved the feeder setup problem based on the assumption that the placement sequence was predetermined or fixed. When the placement sequence is fixed, the feeder setup can be formulated as a quadratic assignment problem (Leipälä and Nevalainen, 1989). Francis et al. (1992) have modelled the feeder setup problem of a turret-type machine as a quadratic assignment problem, since the feeders are assigned to slots on the feeder carriage and the cost of the assignment is impacted by the location of other feeders.

Many researchers have attempted to enhance the feeder setup such as Crama et al. (1990), De Souza and Lijun (1994), Dikos et al. (1997), Foulds and

Hamacher (1993), Grotzinger (1992), Ji et al. (1992, 1994), Leipälä and Nevalainen (1989), Leu et al. (1993), Moyer and Gupta (1996a, 1996b), Sadiq et al. (1993), Sohn and Park (1996) and Sun et al. (2004). Most researchers highlight the fact that the crucial moves, which are subject to optimisation are the feeder carrier movements (Ahmadi and Mamer, 1999; Grotzinger, 1992; Kumar and Luo, 2003), which is the case for moveable feeder carriers. Therefore, optimising the feeder setup, which can lead to optimisation of the feeder carrier movements, is also a crucial factor when optimising the machine throughput.

Foulds and Hamacher (1993) modelled the feeder setup problem as a bin location assignment that was formulated as a single-facility location problem.

By adopting a GA approach, Sun et al. (2004) successfully assigned component feeders to slots by maximising simultaneous pickups in order to minimise the number of pickups that consequently improved machine efficiency. The algorithm evenly allocated the component feeders to the two feeder carriers. They observed that there are empty slots between feeders to maximise simultaneous pickups; the feeders are close to each other so as to minimise the pickup travelling time; and the feeders are located close to the centre position of each feeder carrier such that the robot travelling time between the feeder carrier and the PCB point is minimised.

3.5.5 Component Retrieval Plan Optimisation

If the feeder carrier slots hold several component feeders of the same type (feeder duplication), a decision has to be made on which feeder slot the component type should be retrieved by assuming that a feeder setup and a component pick-and-place sequence have been determined (Bard et al., 1994; Crama et al., 1996). This is the component retrieval problem. The decision is heavily dependent on the modus operandi of the SMD placement machine (Crama et al., 2002). Bard et al. (1994) found a strong relationship between the feeder setup and the component retrieval problem.

Feeder duplication can significantly contribute to the machines throughput (Crama et al., 1996 and 1997; Chen and Chyu, 2003). Some other works that consider feeder duplication are Ahmadi et al. (1988), Bard et al. (1994), DePuy et al. (2000), Francis et al. (1994), Kazaz and Altinkemer (2003), Klincewicz and Rajan (1994) and Ong and Khoo (1999).

Bard et al. (1994) employed a forward dynamic programming approach to solve the component retrieval problem. The optimal component retrieval plan was searched using a branch and bound algorithm. However, Crama et al. (1996) claimed that they invalidated the forward dynamic programming approach proposed by Bard et al. (1994). Consequently, Crama et al. (1996) introduced a two-phase polynomial-time dynamic programming algorithm for solving the component retrieval problem. They viewed the component retrieval problem as a longest path minimisation problem in a PERT/CPM-like network and alternatively as a shortest path problem with side constraints.

3.6 Models and Heuristics

Since the optimisation of the SMD placement machine is a machine specific approach, this section surveys the relationships between machine technologies, models and heuristic methods.

3.6.1 Models and Heuristics for Dual-delivery Surface Mount Device Placement Machine

Since the two gantries cannot access the PCB simultaneously (as they could collide), their pick-and-place operation should be properly scheduled (Sun et al., 2004). To avoid collision, a gantry that completes the picking operations should wait until the other gantry finishes the placement operation, and vice versa. Unlike the other types of SMD placement machine, the efficiency of dual-delivery SMD placement machine, is significantly determined by the gantry workload as well as the gantry scheduling (Sun et al., 2004). However, in the Motorola factory, the gantry workload was not necessarily balanced since the same feeder setup was used for both feeder carriers of every machine (Tirpak et al., 2000). The assembly cycle time, CT, can be computed as a sum of the maximum durations of the pick-and-place cycles between the two heads (Tirpak et al., 2000).

Ahmadi et al. (1988, 1991) emphasised exploiting the concurrency operations in solving this machine problem. The proposed approach attempts to assign the components to the slots in order to balance the workload of both heads as well as minimising the nozzle changes. Due to the extensive setup time of the feeders, they paid more attention to feeder setup time (feeder changeover time) by assigning components to slots when producing many PCBs with many different board types. However, the recent advancement of the SMD placement machine technology has diminished the importance of feeder setup times (Bard et al., 1994).

Since feeder movement is a critical issue for improving the performance of this machine, Ahmadi et al. (1988, 1995) and Grotzinger (1992) both addressed this problem in their work. They identified a hierarchical framework consisting of three optimisation problems; component allocation and partitioning, component pick-and-place sequence, and feeder setup.

Safai (1996) firstly balanced the assignment of the placement points to both heads. They eliminated head contention by assigning each nozzle that has no duplication in the tool bank, to only one of the heads. Their decision to assign the components to the nozzle of both heads is made using a greedy approach so as to minimise the total assembly cycle time including the nozzle change cost. They argued that the solution's quality produced by the approach was superior to the human expert's solution.

An Adaptive Simulated Annealing algorithm has successfully been applied by Tirpak et al. (2000) to solve the feeder, nozzle and placement optimisation problems for the Fuji NP-132. Each iteration of the algorithm requires two main steps: generate a candidate solution and determine if the solution is accepted. Each candidate solution includes a nozzle setup, a feeder setup, and a placement sequence for the two heads. Cheapest insertion and nearest neighbour path construction heuristics are used to construct a placement sequence. A constraint satisfaction swapping heuristic is applied to generate feeder and nozzle setups. The results tested in a Motorola factory show a 3%-12% improvement over the original assembly times.

More recently, Sun et al. (2004), employed a genetic algorithm to simultaneously solve the problem of component allocation and feeder setups in the context of a single machine problem. In order to maintain the consistency in evaluating the solution quality at each iteration, they solved the other decisions (i.e. component pick-and-place sequencing, gantry scheduling etc.) in a simple manner as possible. Indeed, the simultaneous pickups and the number of pickups, that are crucial for assembly cycle time reduction, do not rely on the component placement sequencing, gantry scheduling etc. (Sun et al., 2004). Therefore, the fitness function of the chromosome (i.e. feeder setup) is represented by the maximum workload of the two gantries. Sun et al. (2004) estimated the gantry workload based on realistic move and access times for balancing the gantry workloads whereas other works on multi-station or multi-gantry multi-head machines, only took a summation of a standard mounting time to balance the workload. They observed that the combination of roulette wheel selection and cycle crossover (CX) is the most effective compared to ranking-CX, roulette-PMX (roulette and partially mapped crossover) and ranking-PMX. The experiment on real datasets showed that the proposed algorithm was capable of producing an acceptable quality solution. However, as the GA requires a heavy computation time, it is more realistic to use it in the approach off-line mode rather than on-line mode.

3.6.2 Models and Heuristics for Turret-type Surface Mount Device Placement Machine

Many heuristics/meta-heuristics have been successfully applied in the optimisation of the turret-type placement machines; such as genetic algorithms (Ho and Ji, 2003; Khoo and Loh, 2000; Leu et al., 1993) and greedy approaches (Ellis et al., 2001; Klomp et al., 2000; Kumar and Luo, 2003).

Since the PCB table moves simultaneously and independently in X and Y-directions, the chebychev distance (i.e. $\max(|\Delta x|,|\Delta y|)$ where $|\Delta x|$ and $|\Delta y|$ are the distances between two points in X-coordinate and Y-coordinate, respectively) can be used to determine PCB table movement time (Francis et al., 1992). The turret rotation time is dictated by the component with the slowest turret rotation rate since larger and heavier components are more difficult to hold in place by the suction nozzle and must move slower (Ellis et al., 2001). Due to the various moving parts of the turret-type machine, which has different speeds, Leu et al., (1993) suggested that the total assembly cycle time, CT, should be used as the objective function in solving the problem. In fact, this is also applicable to the other machine types since the machine throughput is a function of CT. The time taken to assemble each component is dictated by the maximum time of the PCB table movement, turret rotation or feeder movement (Leu et al., 1993). In fact, due to the various moving parts of the turret-type machine, the coordination is a crucial factor (Moyer and Gupta, 1996b). The following coordination is required between:

1) Turret rotation and feeder positioning.
2) PCB table movement and turret rotation.
3) PCB table movement along X and Y directions.
4) Component pickup and component placement.
5) PCB table movement and feeder positioning.

Since the turret rotation is an unavoidable movement, Leu et al. (1993) argued that the optimal solution is achieved if the CT is only dictated by the turret rotation movement time.

A two-link GA was devised by Leu et al. (1993) to simultaneously optimise the component pick-and-place sequence and feeder setup of the turret-type machine. Leu et al. (1993) defined a sequence of genes as a link. For a PCB (printed circuit board) having N components, they represented the placement/insertion sequence as a list of numbers between 1 to N. The first link represents the assembly sequence whilst the second link represents the feeder arrangement. Four genetic operators were applied to each link: crossover, inversion, rotation and mutation. Leu et al. (1993) used a total assembly cycle time as a fitness function, with the aim being to minimise the assembly cycle time. They argued that the solution found was almost optimal.

De Souza and Lijun (1994, 1995) incorporated a knowledge-based component placement system with a TSP algorithm to solve the component pick-and-place sequencing problem for a turret-type SMD placement machine. The algorithm first groups the components by type, then by a quantity threshold and finally by the device size. They found that their approach is more practical and superior to the machine vendor software, and obtained a 24% improvement of the board travel distance after applying their approach to the machine generated sequence.

By formulating a feeder setup as a Quadratic Assignment Problem (QAP), Moyer and Gupta (1996a) proposed two heuristic approaches to optimise the feeder setup. The first heuristic is a constructive heuristic that assigns component feeders to slots based on the switching between component types according to the predetermined component placement sequence. The second heuristic is an improvement heuristic that seeks for better assignments by exchanging pairs of slots. They aim to minimise the feeder travelling distance. They obtained better feeder setup compared to Leu et al. (1993) in terms of feeder travelling distance saving. However, the approaches do not necessarily reduce the assembly cycle time since the assembly cycle time is dependent on both the feeder setup and the component pick-and-place sequence. By focusing on reducing the feeder travelling distance, they only reduce the time required for feeders to supply the required components to the turret head. Of course this will help in minimising the CT if the feeder movement time is dominating. Unfortunately, this is not the case since the PCB X-Y movement is the determining factor (in most cases) of the throughput rate of turret-type placement machine compared to the turret rotation time (Gastel, 2002).

As an extension, Moyer and Gupta (1996b) applied the Acyclic Assembly Time (AAT) algorithm to simultaneously improved the quality of the component pick-and-place sequence and feeder setup. The aim of the AAT algorithm is to generate a placement sequence and feeder setup that exploits the unique characteristics of the turret-type machine. In the case where the PCB is still moving to locate the proper placement point, the AAT model allowed the other mechanism to advance to the next position rather than keep idling. Again, Moyer and Gupta (1996b) argued that on average, their approach is superior to Leu et al. (1993) and De Souza and Lijun(1994).

Leipälä and Nevalainen (1989) treated the component insertion sequence as a three-dimensional asymmetric travelling salesman problem whilst the feeder setup was formulated as a quadratic assignment problem.

By expanding the heuristic developed by Leipälä and Nevalainen (1989), Sohn and Park (1996) simultaneously improve the component pick-and-place sequence and feeder setup of this machine type. The component feeders were assigned to slots based on a frequency of use, and then a pick-and-place sequence is determined by also considering feeder setup.

Klomp et al. (2000), viewed a feeder (and its corresponding cluster i.e. set of locations served by a single feeder) as a node in a complete graph. Computational results showed that the gap between the solution found and the lower-bound is relatively small (about 20% in the three machine case), which implies that much of PCB table and feeder rack movements fall within the turret rotation time.

Khoo and Loh (2000) employed a genetic algorithm (GA) to generate the component placement sequence and feeder setup by formulating it as a multi-objective optimisation problem. The prototype system has demonstrated the ability to generate a component placement sequence and feeder setup slightly better than Leu et al. (1993).

Ellis et al. (2001) have developed heuristics for feeder setup and component placement sequencing by using a constructive heuristic that groups together the components with similar PCB table speed and turret rotation speed. The constructive heuristic uses a surrogate function, which can provide a method to approximate penalties for feeder carriage movements, changes in turret rotation speed and changes in PCB table speed. After the initial feeder setup and placement sequence have been constructed, a two-opt heuristic is applied to search for improvement in placement time. Results indicate that the solutions are close to the lower bound and the computational time required to generate the initial solutions is minimal (less than 3 minutes). However, the computational time to generate the improved solution is high, and increases as the problem size increases. For instance, the initial solution computation time is 2 seconds whilst improvement solution requires 1,586 seconds for the smallest PCB. Larger

PCBs requires 164 seconds to generate initial solution and 43,200 seconds to compute the improved solution.

Kumar and Luo (2003) viewed the placement sequencing problem on a Fuji FCP-IV, as a "*generalised Travelling Salesman Problem (TSP)*" where not only the overall travel time depends on the travel sequence (as in standard TSP), but even the distances between any pair of nodes is sequence dependent (whereas in the standard TSP such distances are constant regardless of travel sequence). Since feeder carriage movement is considerably slower than the PCB table movement and turret rotation, and furthermore, these three operations occur concurrently, the whole process is dependent upon the feeder carriage movement. They also show that an optimal tour for the distance matrix provides a desired optimal placement sequence (for a given slot assignment) such that it visits all components of the same part number consecutively. If switching components is required, then the feeder carriage should be moved to the adjacent feeder slots in order to obtain the optimal solution. They show consistent improvement of over 25% in overall assembly time compared to the solution generated by the SMD placement machine optimiser at Lexmark, Inc. For some cases, the rotation of the turret, which takes fixed time, determines the travel time, and thus implies that their optimisation algorithm will be more efficient on machines with faster turret rotation or with smaller rotation angles. However, Kumar and Luo (2003) overlooked the feeder transportation time. If the feeder transportation time is longer than the turret rotation time, then the optimal solution does not hold if all components of the same part number are placed consecutively. Moreover, in some SMD placement machines, the feeder transportation is longer than the time taken for the feeder carrier to move to the adjacent feeder slot.

Ho and Ji (2003) introduced a Hybrid Genetic Algorithm (HGA) integrated with three heuristics to solve the component placement scheduling and feeder setup problems for a chip shooter placement machine. Their genetic algorithm represents a chromosome as two-link structures. The first link represents the sequence of the component placement whilst the second link represents the feeder setup. The initial chromosomes (i.e. initial solutions) are generated using a nearest neighbour heuristic for the first link whilst the second link is randomly generated. During the initialisation stage, each chromosome is improved using an iterated swap procedure and a 2-opt local search. The iterated swap procedure is performed on the first link of each initial chromosome generated by the nearest neighbour heuristic as well as each offspring produced by the genetic operators. A 2-opt local search heuristic is applied to the second link in order to improve the feeder setup of each initial chromosome or offspring generated by the genetic operators. The fitness function represents the total assembly cycle time. Roulette wheel selection is used to select chromosomes to undergo genetic operations. The HGA used a modified order crossover operator and two mutation operators; a heuristic mutation and inversion mutation. Ho and Ji (2003) argued that the HGA is superior to a simple GA used by Leu et al. (1993). They obtained better initial solutions, better final solutions with smaller population sizes and fewer iterations compared to Leu et al. (1993).

Other works, which report improving the turret-type SMD placement machine includes Ellis et al. (2002), Ng (1998, 2000) and Ong and Tan (2002).

3.6.3 Models and Heuristics for Multi-Station Surface Mount Device Placement Machine

Due to the constraints that each station in the multi-station SMD placement machine works concurrently and all stations share the common conveyor system, the synchronisation between conveyer step cycles is the most crucial factor for optimising the machine throughput (Csaszar et al., 2000a). The assembly cycle time, CT, of the machine can be computed as the sum of the maximum completion time of stations in each conveyer step (Csaszar et al., 2000a). Based on the fact that the robotic arm can move simultaneously in the X-Y axes, Csaszar et al. (2000a) use a chebychev distance (refer to 3.6.2) for calculating the CT where the CT is proportional to the robot travelling distance. They observed that in most cases, the seek time (the time taken by the robot to travel between PCB and feeder carrier) is solely dependent on the Y-coordinate of the recent placement point.

As discussed in Section 3.4.2, Shih et al. (1996) employed a simple descent search algorithm to optimise the component pick-and-place operation of a multi-station SMD placement machine. They also employed an expert system approach that focussed more on optimising nozzle switching rather than the component pick-and-place operation.

Subsequently, a genetic algorithm (GA) approach was utilised by Wang et al. (1999) to optimise the feeder setup for the Fuji QP-122, a multi-station SMD placement machine. A penalty function was employed to deal with the machine constraints. They found that the quality of the solution relies more on grouping a set of unique components in the same station, instead of ordering the components in the slots. Therefore the PMX (Partially Mapped) crossover that preserves the information of a group elements has shown good performance. Elitist and tournament selection methods both perform well. By comparison with other optimisation methods, such as a human expert, vendor software, expert system and local search, they found that a genetic algorithm is a suitable approach for solving the problem.

Csaszar et al. (2000b) employed a knowledge-based system to optimise the multi-station machine, which has a single head and a single nozzle per station. The system was designed to emulate human experts. They divided the allocation problem into two sub-problems. These were the assigning of components to the stations, and the arrangement of components within the stations (feeder setup) to achieve maximum throughput by minimising the head movements. The expert system was split into four phases: simulator pre-processing, placement, refining and conversion phases. The results show that the expert systems uses an average of 16.14% fewer feeder slot than the vendor's software and the throughput improved by 13.47% to 15.95%. They show that by using the cost function of the number of placements together with pick-and-place time they gain better results than using just the pick-and-place time.

In other work, Csaszar et al. (2000a) extended their own work of optimising the same machine type by utilising a tabu search algorithm. Since the machine has many stations, they solved the problem of allocating components to the stations, feeder setups and component pick-and-place sequencing problem for each station. They partitioned the problem into two phases and solved them using a tabu search and a specific heuristic, respectively. Unfortunately, they do not explicitly explain how the original problems are solved using their proposed approach. Indeed, the results presented were insufficient to evaluate the effectiveness of their approach in solving the multi-station SMD placement machine.

To date, there have not been many works, which have focussed on optimising multi-station SMD placement machine. This is probably due to the fact that the machines are complex and present many optimisation challenges, especially when all the various problems that need to be solved are considered. In short, some of the heuristics that have been applied to solve the multi-station SMD placement machines are expert system (Csaszar et al., 2000b; Shih et al. 1996), genetic algorithm (Wang et al., 1999) and tabu search (Csaszar et al., 2000a).

3.6.4 Models and Heuristics for Multi-Head Surface Mount Device Placement Machine

A lot of work has been carried out to optimise the component pickup-and-placement sequence of the multi-head SMD placement machine such as Altinkemer et al. (2000), Burke et al. (2001), Crama et al. (1990 and 1997), Ho and Ji (2004), Jeevan et al. (2002), Van Laarhoven and Zijm (1993), Lee et al. (1999) and Magyar et al. (1999).

Lee et al. (1999) developed heuristics, which are based on dynamic programming and the nearest neighbour TSP to solve the optimisation of multi-head SMD placement machines. They chose a hierarchical method to find the solution by starting with the construction of reel-groups, then the assignment of reel-groups and finally the sequencing of pick-and-place movements. Since nozzle changes are time-consuming and the number of nozzle changes is proportional to the number of nozzles to be used, they choose a nozzle for each reel in such a way that the total number of nozzles is minimised. Then, they assign the reels to heads in such a way that each head has about the same workload. Since nozzle changes are the most time-consuming operation, they decided to first determine the order of nozzle changes before determining the sequence of pick-place movements. The simulation results indicate that their method achieves an average saving of 18% in PCB assembly time over the heuristic algorithm supplied by Yamaha.

Magyar et al. (1999) tackled the problem of determining the sequence of component pickup-and-placement; scheduling the assignment of different nozzles (tool) to the robot head; and feeder setup by adopting a hierarchical problem solving approach. They studied the problem of the GSM machine (Universal, 1999) that is a multi-head SMD placement machine that has one placement head equipped with four pipettes (or spindles) and each of them can handle one component. Firstly they solved the feeder setup problem by using a greedy local search that searched for increasing the number of gang-pickup (i.e. a simultaneous pickup where many components are picked up at a given time). The output of the feeder setup is given as an input for a nozzle optimisation procedure whilst the output of the nozzle optimisation procedure, served as input to the component pick-and-place procedure that also employed a greedy local search heuristic. Their system significantly improved the assembly cycle time when tested for real industrial products.

Since the arm and head can move simultaneously in both the X and Y directions, Altinkemer et al. (2000) used the chebychev distance (refer to 3.6.2) and calculated the distance as the maximum of the movements in the X and Y direction. They consider two cases; when the feeder locator moves and when the feeder locator does not move. When the feeder locator moves, the feeder of the component type, that will be processed next can move towards the tool magazine while the head is mounting another component type, so the distance between the feeder locations and the points on the PCB can be measured from a fixed point next to the tool magazine. The simultaneous movement enables each component type to have the same origin and destination point, and thus allow the formulation to be an independent capacitated vehicle routing problem (VRP). Since the distance between a point on the PCB and feeder is not dependent on where the component is located among feeders, the feeder setup problem does not have to be integrated with the pick-and-place sequencing decisions. For the case where the feeder locator does not move, they formulate the problem as a combination of assignment-like and vehicle-like problems. They first solve a VRP for each component type at every possible feeder location, and then use this feasible solution as the cost of assigning the component type to the particular feeder location. They argue that their integrated algorithm provides a feasible solution with an error gap less than or equal to the maximum error gap of the VRP costs.

Burke et al. (2001), have introduced a three phase heuristic to deal with the assembly of multiple printed circuit board types with different batch sizes on a single machine without set-ups between board types. Experimental results show that their approaches are very promising.

Jeevan et al. (2002) employed a genetic algorithm to optimise the component pickup-and-placement of the multi-head SMD placement machine. They represent a distance of a TSP tour (i.e. a total pickup-and-placement distance) as a fitness function. However, they do not explicitly discuss their mathematical model and chromosome representation in their paper.

More recently, Ho and Ji (2004) applied the same approach that was introduced in (Ho and Ji, 2003) to solve the component placement scheduling and feeder setup problems for a multi-head placement machine. In solving a multi-head placement machine, Ho and Ji (2004) claimed that their approach also outperformed a simple genetic algorithm used by Ong and Khoo (1999) in terms of the total travelling distance of placement head.

3.6.5 Models and Heuristics for Sequential Pick-and-Place Surface Mount Device Placement Machine

Ball and Magazine (1988) formulated the placement sequence problem as a type of directed postman problem. They show that the balance and connect heuristic can be applied to this problem.

Kumar and Li (1995) model the optimisation of feeder setup and component pick-and-place sequence for a sequential pick-and-place SMD placement machine as an instance of a linear integer programming problem. They solve the problem by determining an assignment of pickup slots and a component assembly sequence for each individual nozzle. Heuristics such as nearest neighbour, nearest insertion, furthest insertion, and random generation are used to construct an initial assembly sequence, and the other heuristics such as 2-opt and 3-opt are used to improve upon the initially generated assembly sequence. Simulation results show a consistent assembly time saving of 25% over the current approach used in the factory.

Ahmadi and Mamer (1999) have modelled the problem of sequencing the part types for placement and the problem of scheduling the movement between points on the PCB as a collection of interdependent travelling salesman problems. The computational results show that the approximation of the problem by a sequence of TSPs was able to produce significant increases in throughput.

Ong and Khoo (1999) employed a genetic algorithm approach to simultaneously solve the component pick-and-place sequencing and feeder setup problems. The objective function, which represents a fitness function, was to minimise the travelling distance of the placement head. They applied the two-link genetic algorithm proposed by Leu et al. (1993) to optimise the sequential pick-and-place SMD placement machine. They also addressed the advantage of allowing feeder duplication.

Fu and Su (2000), Hop and Tabucanon (2001a, b), Su et al. (1995), Wang et al. (1998) strongly believe that robotic travel routing should be based on relative coordinates to obtain a better solution because the robotics, board and magazine are simultaneously moved at different speeds during assembly. Their dynamic pick-and-place (DPP) model has been introduced by Su et al. (1995). In the DPP model, the robot moves vertically along the Y-axis (in the optimal condition), while the PCB table and feeder rack move horizontally along the X-axis, and the pickup-and-placement point are dynamically allocated. The optimal condition occurs when the robot travels only in the Y direction, and no movement in the X direction is observed (Wang et al., 1998). They modelled the sequential pick-and-place SMD placement machine. Details of these works are further discussed in chapter 4 (see section 4.2).

3.6.6 Heuristic for Unclassified Surface Mount Device Placement Machine

There are some works that do not mention the characteristics of the SMD placement machine being solved such as Chang and Terwilliger (1987) and Khoo and Ong (1998).

Chang and Terwilliger (1987) proposed a rule-based approach to solve the component placement sequence problem by considering the PCB layout to avoid the interference problem. The first two rules sequence the components to be mounted by a mounting head (nozzle) in the direction of the mounting head’s largest clearance area requirements and in a path that takes minimum time. They also sequence the mounting heads within an assembly station in order of descending clearance area required by each mounting head and tried to minimise the nozzle changes. The last rule sequences the assembly stations in order of descending clearance area required by its nozzle(s). To resolve the conflict among the rules, Chang and Terwilliger (1987) used the ranking approach. No results were reported.

Khoo and Ong (1998) demonstrated the possibility of applying a genetic algorithm for optimising printed circuit board assembly planning. They incorporated three of the four (Sanchez and Priest, 1991) basic insertion/sequencing rules (they excluded the third rule, i.e. assemble components of identical sizes and shapes, and then assemble the other components of non-similar size and shape) into the algorithm. Order-based crossover, inverse mutation and dual mutation were found to be suitable for solving the problem. Khoo and Ong (1998) applied a polygamy mechanism to further enhance their genetic algorithm approach. They gained a 24.28% reduction in total PCB table travel distance compared to the initial solution. However, due to the concurrency mechanism of the SMD placement machine, reducing the PCB table travel distance does not guarantees maximisation of the machine throughput.

3.7 Summary

This chapter presented a survey of single surface mount device (SMD) placement machine optimisations. By combining the sub problems addressed by Crama et al. (2002) and Magyar et al. (1999), this work classified the single SMD placement machine problem into five sub problems, these being a feeder setup, a component pick-and-place sequencing, a component retrieval plan, a motion control and a nozzle optimisation. These sub problems are tightly intertwined. Consequently, some researchers solved the problem in an iterative manner, instead of a one-pass procedure through each of the sub problems, and some used an integrated approach. Nevertheless, some works have addressed the sub problems independently by making assumptions about the rest of the sub problems.

There have not been many researchers who have reported improving the motion control. This might be because this decision is directly relevant to the production preparation (Van Laarhoven and Zijm, 1993). In fact, many SMD placement machines use fixed pick-and-place points since not many of them have moveable head (X-Y), moveable feeder carriers and a moveable PCB table.

Many papers defined the component pick-and-place optimisation as finding a shortest route to pick-and-place the electronic components onto the PCB. This is only true if other factors such as nozzle switching, feeder transportation time (i.e. time taken by the feeder to transport the component to the pickup position), gang pickups (i.e. simultaneous pickup) etc. are ignored. Therefore, it is more precise to define the component pick-and-place optimisation as finding a shortest time to pick-and-place the electronic components onto the PCB.

Generally, most researchers modelled the component pick-and-place optimisation as a TSP problem. The PCB points are defined as cities whilst the time between components placements represent the distance among cities. Unfortunately, the time between components placements relies on many factors such as nozzle changeover, component feeder transportation, the acceleration forced on the pre-mounted component (for the case of movable PCB table), a components grouping in a sub tour (for the case of the machine, which has many pipette/nozzle in a head), etc. Therefore, many researchers abstracted the component pick-and-place optimisation as a TSP problem by representing the head travel distance as a distance among cities.

The evaluation of the solution quality of the feeder setup is not straight forward as we need other interrelated decisions such as nozzle assignment and sequencing, pick-and-place sequencing, gantry scheduling etc. Nevertheless, the other decisions could be solved in order to avoid disturbances and maintain the evaluation's consistency while searching for an improved feeder setup. Therefore many works solved the feeder setup by assuming the other sub problems were determined. By fixing the component pick-and-place sequence, most works formulated the feeder setup as a quadratic assignment problem.

So far, not many works have focussed on solving the component retrieval plan problem. The problem might be indirectly solved while determining the component pick-and-place sequence.

When the SMD placement machine has more than one nozzle per head (or even a single nozzle per head), choosing an effective nozzle group (or a nozzle) is important in order to improve the pick-and-place operations and to minimise the number of nozzle changeover operations. Having a proper nozzle group assignment might lead to having more simultaneous pickup operations, minimising feeder carrier movement, robot arm and/or PCB table movements, that can ultimately improve the machine throughput. Optimising the pick-and-place operation without considering the nozzle switching operations, may not be efficient since it may cause many unnecessary nozzle changes that will significantly reduce the machine throughput. Since the nozzle changeover operation is very time consuming, the nozzle optimisation can be considered as the most important factor when improving the machine throughput. Unfortunately, very little work has addressed the optimisation of the frequency of nozzle changeover operations.

Since there are various types of SMD placement machines, which have different characteristics and restrictions and the PCB production scheduling process is highly influenced by the type of SMD placement machine being used, this chapter has also attempted to classify the SMD placement machine based on the specification and operational methods. The SMD placement machines may be arranged into five categories: dual-delivery, multi-station, turret-type, multi-head and sequential pick-and-place.

The survey associated the models, assembly machine technologies and heuristic methods. Genetic algorithm approaches have been applied to optimise all types of SMD placement machines. Knowledge-based systems are also applicable for solving some type of SMD placement machine such as turret-type and multi-station. Tabu search, simulated annealing and integer programming are rarely used in solving SMD placement machine. As far as we are concerned, none of the research in this field has reported applying variable neighbourhood search and hyper-heuristic approaches. This is, as yet, an unexplored research area in this field. Due to a complexity of the problem, which involves many machine constraints, most research in optimising the SMD placement machine utilised a greedy search heuristic, which is very problem specific.

As the optimisation of the SMD placement machine is very machine specific, this work strongly suggests that researchers clearly define the machine characteristic and operational methods. For an evaluation and comparable purposes, this work also suggests that researchers clearly define their objective function, which is usually not very clearly stated in many of the reported works in this field. It is more precise to formulate the main objective function in terms of optimising the assembly cycle time, CT, instead of optimising the head travel distance, PCB travel distance, feeder carrier travel distance, etc. since the machine throughput is a function of the CT. Moreover, due to concurrency operations, optimising one of the movements does not guarantee optimisation of machine throughput. Indeed, many other determining factors are involved in determining the efficiency of the SMD placement machine such as nozzle optimisation, component feeder transportation etc.

The next chapter presents a dynamic pick-and-place point specification approach for improving the motion control specification approach. The chapter proposes a revised dynamic pick-and-place point (DPP) specification approach called Chebychev DPP. The chapter also introduces a triple objective function that aims to minimise the CT, PCB table movements and feeder carrier movements for improving a feeder setup.

Chapter 4

Dynamic Pick-and-Place Point Specification Approach

4.1 Introduction

Some factors that significantly contribute to the overall assembly efficiency of the surface mount device (SMD) placement machines are the robot motion control, the sequence of pick-and-place points and the feeder setup. Many techniques have been developed to improve the sequence of placement points and/or the feeder setup for the printed circuit board (PCB) assembly process (these were reviewed in chapter 3). However, only limited work has been carried out on improving the robot motion control.

Moreover, most of the published work only aims to minimise the assembly cycle time. The movement of the feeder carrier and the PCB table are not normally factors, which are minimised. In the case where the feeder carrier and the PCB table are moveable, we should also consider minimising their movement. Hence, our work proposes a new objective function that attempts to minimise the assembly cycle time together with the minimisation of the movement of the feeder carrier and the PCB table. By integrating the three minimisation factors, we can still achieve an assembly cycle time as good as those which only aim to minimise the assembly cycle time. Moreover, we can gain improved feeder carrier movement and PCB table movement. Reducing these movements may also prolong the life cycle of the placement machine even if it does not affect the throughput rate of the machine.

This chapter proposes a revised dynamic pick-and-place point (DPP) specification approach called Chebychev DPP (CDPP). The DPP model attempts to maintain the fixed pickup-and-placement location as much as possible unless this leads to robot idling. Therefore, the DPP model may still have unnecessary movement. Hence, in our CDPP, we try to eliminate the unnecessary movement by looking forward to the next PCB coordinate when determining the current pickup location and looking forward to the next feeder slot when determining the current placement location. Instead of only searching for a minimum assembly cycle time, we are looking for a minimum assembly cycle time and a reduction in feeder carrier and PCB table movement as far as possible. We formulate a problem for an SMD placement machine that is a type of cartesian robot, which has a single head equipped with a single pipette/nozzle, which is able to move in both the X and Y directions concurrently. This machine is a type of sequential pick-and-place machine as classified in chapter 3 (section 3.3.5). The formulations are constructed based on the triple objectives of minimising the robot assembly cycle time, feeder movements and PCB table movements. The main difference between our CDPP model and the previous DPP (and extended DPP i.e. EDPP) is that our CDPP calculates the robot arm movement distance as the maximum of the movement in the Y or X direction (a chebychev distance) since our robot arm can move in X-axis and Y-axis concurrently, whilst the previous DPP (and EDPP) calculate the robot arm movement as a euclidean distance. The work presented in this chapter has been disseminated as follows:

a) Ayob, M. and Kendall, G. (2002b). A new dynamic point specification approach to optimise surface mount placement machine in printed circuit board assembly. *Proceeding of the IEEE ICIT'02*, Bangkok, 11-14 Dec., 486-491.

b) Ayob M. and Kendall G. (2005a). A triple objective function with a chebychev dynamic point specification approach to optimise the SMD placement machine. *European Journal of Operational Research,* 164, 609-626.

4.2 Related Work

Su et al. (1995) proposed a dynamic pick-and-place (DPP) point to avoid robot idling time. This approach allows the robot to pick-and-place a component at any location rather than fixed pickup-and-placement (FPP) locations. The pick-and-place sequence was determined using a travelling salesman problem (TSP) method and the feeder slots were randomly arranged. They found that the DPP approach was superior to the FPP approach. The works on DPP, was subsequently followed by Wang (1996), Wang et al. (1995, 1997, 1998), Fu and Su (2000) and Hop and Tabucanon (2001a, b).

Wang (1996) developed three setups to allow the dynamic allocation of pick-and-place points, these being one-magazine-and-one-board (1M1B), one-magazine-and-two-board (1M2B) and two-magazine-and-one-board (2M1B). The PCB points and component types were randomly generated. Wang applied Karg and Thompson's algorithm (Karg and Thompson, 1964) to determine the component pick-and-place sequence whilst the feeder setup (allocating components to feeder slots) was arranged based on Wang et al.'s algorithm (Wang et al., 1997) that was published as a technical report (Wang et al., 1995). Experimental results showed that the 2M1B and 1M2B setups are more efficient than the 1M1B setup.

Wang et al. (1997) proposed a heuristic approach for feeder setup integrated with DPP robot motion control to improve SMD placement machine efficiency by assuming that an assembly sequence already exists. Based on the principle that minimising the number of robot interceptions (i.e. the robot intercepts with the feeder carrier or the PCB table) can lead to a reduction in assembly cycle time, they assigned components to feeder slots such that the total exchange frequency of all adjacent slot pairs have the maximum value. The exchange frequency is defined as an index that counts the exchange frequency between two different component types for succeeding pickups. For example, if the i^{th} placement sequence involves *g* component types followed by *z* component types for the $(i+1)^{th}$ placement sequence, the exchange frequency between component type *g* and *z* is counted as '1' ($F_{gz}=1$). Wang et al. (1997) converted a feeder setup problem into a travelling salesmen problem by representing the slots as nodes and the exchange frequency between each slot pair as the length of an arc connecting the two nodes. In order to convert the problem of feeder setup into a travelling salesman model, Wang et al. (1997) subtracts all the exchange frequencies between each slot pair from a large number (a number larger than all exchange frequency values). Hence, they arrange the feeder slots by searching for the minimum total exchange frequency. By doing this, Wang et al. (1997) assumed that minimising (or actually maximising) the total exchange frequency can lead to the minimum assembly cycle time (CT).

Next, Wang et al (1998) applied off-line heuristics to improve the placement sequence and feeder setup to take advantage of the DPP model. They first sequence the placement by considering the adjacent placement to be as close as possible (minimum travel in X direction) to achieve minimum total travel while placing each component in its proper location. Next, by adopting a feeder setup heuristic from Wang et al (1997), they organised feeder slots to let all adjacent slots have the highest exchange frequency pickups of the same component type. They claim that the approach provides a rapid solution to even very large problems, and demonstrate the on-line implementation to ensure the feasibility of the approach. Nevertheless, Wang's approach was still based on a fixed coordinate system using the TSP method to obtain robotic travel routing (Su and Fu, 1998; Su et al., 1998).

Su and Fu (1998) simultaneously arranged the placement sequence and feeder slots based on the DPP approach by applying a simulated annealing (SA) algorithm. In other work, Su et al. (1998) applied tabu search to find a good placement sequence and feeder setup. They argued that the robotic travel routing should be based on relative coordinates since the coordinates of the pickup-and-placement points change all the time, i.e. the robot arm, board and feeder carrier move at different speeds. In both works, they obtained better performance when compared to Wang et al. (1997).

Subsequently, Fu and Su (2000) evaluated the effectiveness of a genetic algorithm (GA), simulated annealing (SA) and tabu search (TS) in solving the component placement sequencing and feeder setup problems. In the GA approach, a possible solution is represented by a chromosome, and each gene in the chromosome represents the insertion point and the component. The fitness function is defined as the cycle time of the robotic assembly. In the SA and TS algorithms, the potential solution is represented by a vector. The cycle time of the robotic assembly is defined by the energy function in SA or the objective function in TS. These approaches can simultaneously arrange the insertion sequence and assign the feeder slots and yield a better performance compared to the conventional approach (FPP model). Computational experiments indicate that the GA's performance is the worst among these approaches in a larger number of insertion points and/or component types. A GA requires more computational time for the larger tested cases for a near optimal solution to be obtained. The performance of TS is better than the performance of the other two approaches in problems of smaller sizes. SA is a robust technique that has better performance for the larger problem sizes if the computational time is limited. They gained better performance than Wang et al. (1997).

Recently, Hop and Tabucanon (2001a) attempted to minimise the assembly cycle time together with minimisation of the feeder carrier movement and the PCB table movement. They proposed a new heuristic, which they called a multiple criteria approach and were able to improve on the results of Wang et al. (1998). The approach was based on the fact that assembly time depends on the relative position of pickup-and-placement points (DPP model). The multiple criteria approach uses the trade-off between two strategies:

- Minimise the PCB table travel distance, where the strategy is 'assemble by area'.
- Minimise the feeder rack travel distance, where the strategy is 'assemble by component type'.

The idea is to place the same component types as close to each other as possible on the PCB. The algorithm starts with an initial solution using Wang's method (Wang et al., 1998). Hop and Tabucanon (2001a) construct an initial feeder setup and a component pick-and-place sequence using Wang's method. Then a new feeder setup is constructed based on assembly by component type. Next, the assembly sequence is generated based on the principle of assemble by area. These two steps are iterative, and will be terminated when there is no improvement in assembly time or the number of iterations reaches a pre-defined limit. Computational results indicated that the new approach performed better than the approach by Wang in terms of total assembly time. The reduction in time ranged between 4% for lower density boards, to 28% for higher density boards. It showed that as the board density increases, the chances of a robot arm waiting for the PCB table movement reduces. However, Hop and Tabucanon did not measure the saving of the feeder carrier movement and the PCB table movement. They only hoped that their strategy was able to reduce the PCB table movement and feeder carrier movement.

In other work, Hop and Tabucanon (2001b) proposed an extended dynamic point specification approach (EDPP). The EDPP model determines the pickup-and-placement coordinate on the PCB based on a global view of the point relationships in the system. The EDPP considers the movement of the robot arm, the movement of the PCB table and the movement of the feeder carrier as a way of reducing the assembly cycle time. If the feeder carrier (or PCB table) can move fast enough to position the required point at the required pickup (or placement) location, the EDPP model may allow the feeder carrier (or PCB table) to pass over the required point and stop at the point where the feeder carrier (or PCB table) can provide better robot movement. This means that the EDPP is willing to pay an extra cost for robot travel in order to gain better feeder movement or PCB table movement for the next assembly cycle. Hop and Tabucanon (2001b) formulated three different cases; (1) fixed PCB table, dynamic feeder carrier and robot arm; (2) fixed feeder carrier, dynamic PCB table and robot arm and (3) dynamic PCB table, feeder carrier and robot arm. The EDPP model obtained better assembly cycle time compared to the DPP model designed by Su et al. (1995).

4.3 Surface Mount Device Placement Machine

This work only focuses on a problem of a sequential pick-and-place machine (as classified in chapter 3 section 3.2.5) that has a single arm with a single head equipped with a single pipette/nozzle. The robot (that is the arm and head) is able to move in both X and Y directions concurrently to pick-and-place a component. The placement machine uses a nozzle to grasp a component from the feeder carrier and then mounts it onto the PCB. The feeder carrier and the PCB table are moveable in the X-axis to position the component pickup coordinate and the placement coordinate of the PCB, respectively. The robot, PCB table and feeder carrier can move concurrently. The robot travels between feeder carrier and PCB table for picking and placing a component, respectively.

4.4 Fixed Pick-and-Place Point Model

In the Fixed Pick-and-Place Point (FPP) model, the feeder carrier can move horizontally (along the X-axis) to position a required component at the required pickup location. The PCB table can move, concurrently in the X/Y-axis to position a PCB coordinate at the required placement location. The robot arm can only move in the Y-axis between fixed pickup-and-placement locations (Fu and Su, 2000) (see figure 4.1). Since the robot arm only moves between these two fixed locations, and the speed of the PCB table, the feeder carrier and the robot arm varies and, furthermore, their travelling distances also differs, there may exist an undesirable robot waiting time.

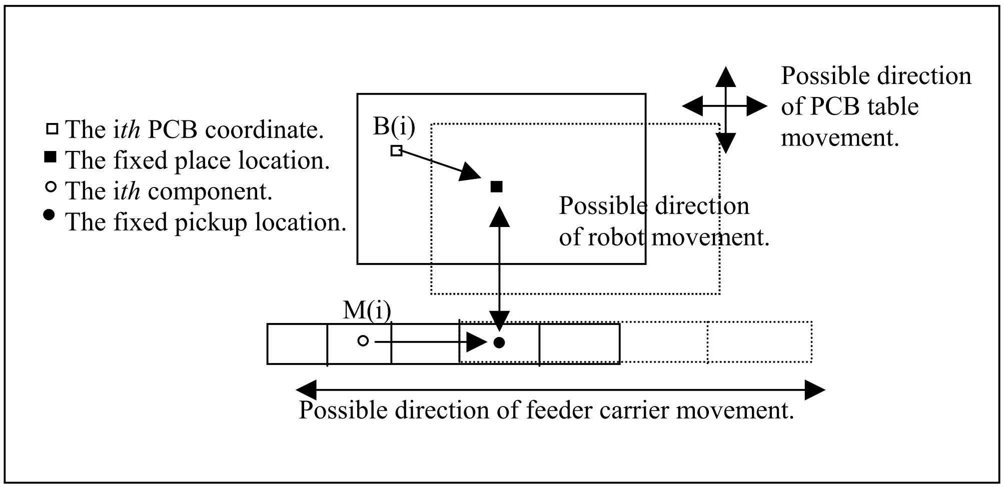

□ The ith PCB coordinate.
■ The fixed place location.
○ The ith component.
● The fixed pickup location.
B(i)
M(i)
Possible direction of PCB table movement.
Possible direction of robot movement.
Possible direction of feeder carrier movement.

Figure 4.1: The FPP model.

4.5 Dynamic Pick-and-Place Point Model

In the Dynamic Pick-and-Place Point (DPP) model, both the feeder carrier and PCB table are still moveable but only in the X-axis, whilst the robot arm moves in the Y-axis, in optimal conditions. This case only happens when the feeder carrier or PCB table can move within 'free' movement time, that is when the feeder carrier and PCB table can move to the best pickup-and-placement point before the robot arm arrives. However, when the robot arm can arrive at the best pickup location before the feeder carrier can bring the required component to the best pickup location, or when the robot arm can arrive at the best placement location before the PCB table can position the required PCB coordinate at the best placement location, then the robot arm has to move at an angle from the Y-axis to catch the feeder carrier or PCB table at a dynamically allocated coordinate to avoid robot idling (Fu and Su, 2000). Both the PCB table and the robot arm, or the feeder carrier and the robot arm, will stop and meet at the dynamically assigned interception location at the same time (Su et al., 1995). This situation is defined as a robot interception. In the DPP model, the robot arm can pick-and-place a component at any location along the feeders and the PCB, respectively. Figure 4.2 and figure 4.3 demonstrate how the pickup-and-placement locations are determined in the DPP model.

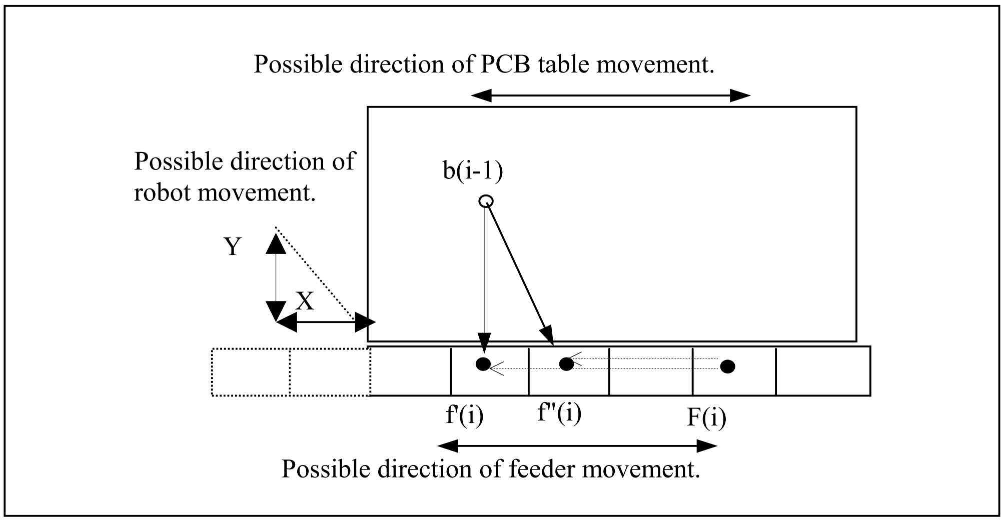
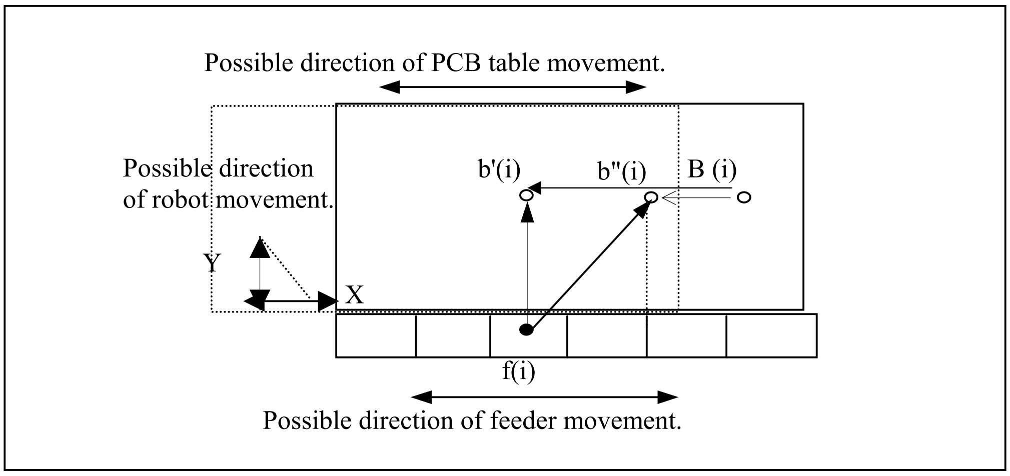

Possible direction of PCB table movement.
Possible direction of robot movement.
b(i-1)
Y
X
f'(i)
f"(i)
F(i)
Possible direction of feeder movement.

Figure 4.2: The DPP model for determining pickup location *f(i)*. The feeder will carry the component from *F(i)* to *f'(i)* if no robot interception occurs or from *F(i)* to *f"(i)* otherwise.



Possible direction of PCB table movement.
Possible direction of robot movement.
b'(i)
b"(i)
B (i)
Y
X
f(i)
Possible direction of feeder movement.

Figure 4.3: The DPP model for determining placement location *b(i)*. The PCB table will move to position the placement point from *B(i)* to *b'(i)* if no robot interception occurs or from *B(i)* to *b"(i)* otherwise.

The following notations are used (most of them adopted from Wang et al., 1998) to describe the DPP and the CDPP models:

CT	: *the cycle time to assemble all components;*
N	: *the number of PCB points;*
K	: *the number of component types (each feeder slot holds one component type only);*
$c(i)_{x,y}$	: *the i*th *X,Y coordinate on the PCB, which will have the i*th *component placed there;*
$f(i)_{x,y}$	: *the feeder pickup coordinate of the i*th *assembly sequence. The* $f(0)_x$ *is defined as the centre of the first pickup location (referred to as the origin coordinate). For all i,* $f(i)_y=0$ *as the feeder slot can only move in the X direction;*
$b(i)_{x,y}$	: *the placement coordinate of the i*th *assembly sequence, which is the X,Y offset from the origin coordinate* ($f(0)_x$). *For all i,* $b(i)_y=c(i)_y$ *as the PCB table can only move in the X direction ;*
V_r	: *the robot speed (average);*
V_b	: *the PCB table speed (average);*

V_f	: *the feeder speed (average);*
λ	: *the time for picking up a component;*
θ	: *the time for placing a component;*
F_{ab}	: *the exchange frequency between components of type a and b;*
Fm(i)	: *the moving distance and direction of feeder (positive sign means the feeder moves to the left, negative otherwise) to position the i*th *component at the i*th *pickup location,* $f(i)_{x,y}$*;*
Tm(i)	: *the moving distance and direction of the PCB table (positive sign means the PCB table moves to the left, negative otherwise) to position the i*th *PCB coordinate at the i*th *placement location,* $b(i)_{x,y}$*;*
$s_{i-1,i}$	: *the slot distance between feeder slot for i*th *and (i-1)*th *component in the assembly sequence (positive sign means the i*th *slot is located at the right side of (i-1)*th *slot, negative otherwise);*
$c_{i-1,i}$	: *the distance between the i*th *and the (i-1)*th *points on the PCB board (positive sign means the X-coordinate of the i*th *point on PCB is bigger than the (i-1)*th *point, negative otherwise);*
$d_{f(i),b(i)}$	: *the distance between f(i) and b(i) where the distance is measured as a euclidean distance in DPP or a chebychev distance in CDPP ;*
D_x	: *the interception distance in X-axis (positive sign means the robot arm moves to the right, negative otherwise).*
B(i)	: *the location of the i*th *PCB coordinate when the robot arm is placing the (i-1)*th *component at b(i-1).*
F(i)	: *the feeder slot location of the i*th *component when the robot arm is picking up the (i-1)*th *component at f(i-1).*

When the robot moves from the *(i-1)*th pickup location on the feeder carrier, *f(i-1)*, to the *(i-1)*th placement point on the board, *b(i-1)*, the feeder carrier concurrently moves to position the next component at the pickup location *f'(i)* on the feeder carrier (refer to figure 4.2). Similarly, when the robot moves from the *(i-1)*th placement point, *b(i-1)*, to pickup the *i*th component at *f(i)*, the PCB table simultaneously moves from the placement point *b(i-1)* to position the *i*th

PCB coordinate, $c(i)_{x,y}$ point, at the best placement location, *b'(i)* (refer to figure 4.3). Since the robot arm, the feeder carrier and the PCB table move at different speeds and vary in the distance they need to travel, the next possible pickup-and-placement locations are dynamically determined to avoid robot idling. If the feeder carrier can move fast enough to bring the i^{th} component from *F(i)* to *f'(i)*, then the next possible pickup location is *f'(i)* where the X coordinates of *b(i-1)* and *f'(i)* are the same and *f(i)=f'(i)* (refer to figure 4.2). Otherwise, the robot arm will pick the i^{th} component when it meets the feeder carrier at the interception location, *f''(i)* and *f(i)=f''(i)*. Similarly, in determining the next placement location, if the PCB table can move fast enough to position the i^{th} PCB coordinate, $c(i)_{x,y}$, from *B(i)* to *b'(i)*, then the next possible placement point is *b'(i)* where the X coordinates of *b'(i)* and *f(i)* are the same and *b(i)=b'(i)* (refer to figure 4.3). Otherwise, the robot arm will place the i^{th} component onto the i^{th} PCB coordinate when it meets the PCB table at the interception location, *b''(i)* and *b(i)=b''(i).* For every pickup-and-placement operation, the movement of the robot arm is unavoidable. Hence, the time taken for the robot arm movement is a major contribution to the total CT (by ignoring the other factors such as machine down time, nozzle change etc.). However, the effect of the movements of the feeder carrier and the PCB table can be eliminated. If the pickup-and-placement sequence are optimally scheduled, and a location of where the robot arm should pick-and-place a component are well designed, then the PCB table and the feeder carrier may move within a free movement time (i.e. they can move within the time taken for the robot arm movement) and their movements do not affect the CT. Hence, one of the aims of our work is to increase the optimal robot movement in order to minimise the assembly cycle time, CT, which is a function of the total robot travelling distance divided by the robot speed, plus the total pickup-and-placement time. Thus the aim is;

$$CT = \min \sum_{i=0}^{N-1} \left[\frac{d_{f(i),b(i)}}{V_r} + \frac{d_{b(i),f(i+1)}}{V_r} \right] + N\theta + N\lambda; \quad \text{when i=N-1, then f(N)=f(N-1)} \qquad \textbf{(4.1)}$$

Wang et al. (1998) claimed that the shortest robot travelling distance occurred when both $d_{f(i),b(i)}$ and $d_{b(i),f(i+1)}$ involve no robot arm movement in the X direction. In the DPP approach, the optimal pickup occurs when equation

(4.2) is true, that is when the total time taken for the robot arm to move from the pickup point *f(i-1)* to the placement point *b(i-1)*, to place the *(i-1)*th component and to move from the placement point *b(i-1)* to the next best possible pickup point *f'(i)*; is greater than the time taken for the feeder carrier to bring the *i*th component from location *F(i)* to the best pickup location *f'(i)* where the best pickup point is the case when $f'(i)_x = b(i-1)_x$.

$$\frac{d_{f(i-1),b(i-1)}}{Vr} + \theta + \frac{d_{b(i-1),f'(i)}}{Vr} \geq \frac{d_{F(i),f'(i)}}{V_f} \qquad \textbf{(4.2)}$$

The optimal placement occurs when equation (4.3) is satisfied, that is when the total time taken for the robot arm to move from the placement point *b(i-1)* to the pickup point *f(i)*, to pick the *i*th component and to move from the pickup point *f(i)* to the next best possible placement point *b'(i)*; is greater than the time taken for the PCB table to bring the *i*th placement point from location *B(i)* to the best placement location *b'(i)* where the best placement location is the case when $b'(i)_x = f(i)_x$.

$$\frac{d_{b(i-1),f(i)}}{Vr} + \lambda + \frac{d_{f(i),b'(i)}}{Vr} \geq \frac{d_{B(i),b'(i)}}{V_b} \qquad \textbf{(4.3)}$$

When equation (4.2) does not hold, that is when the robot arm can reach point *f'(i)* before the feeder carrier can arrive at point *f'(i)*, then instead of moving in *Y* direction from the *b(i-1)* to *f'(i)* and waiting for the feeder carrier at *f'(i)*, the robot arm will move at an angle from the *Y-axis* from *b(i-1)* to pick the *i*th component at the interception location, *f''(i)*. Both, the robot arm and the feeder carrier, will meet at *f''(i)* and stop moving at the same time. This condition is represented by equation (4.4).

$$\frac{d_{f(i-1),b(i-1)}}{Vr} + \theta + \frac{d_{b(i-1),f''(i)}}{Vr} = \frac{d_{F(i),f''(i)}}{V_f} \qquad \textbf{(4.4)}$$

Similarly, when equation (4.3) does not hold, the robot arm will move an angle of *Y* from the *f(i)* to place the *i*th component onto the *i*th PCB coordinate at the interception location, *b''(i)*. Both, the robot arm and the PCB table will meet at *b"(i)* and stop moving at the same time. This condition is represented by equation (4.5).

$$\frac{d_{b(i-1),f(i)}}{Vr} + \lambda + \frac{d_{f(i),b''(i)}}{Vr} = \frac{d_{B(i),b''(i)}}{V_b} \qquad \textbf{(4.5)}$$

4.6 Chebychev Dynamic Pick-and-Place Point Model

The Chebychev Dynamic Pick-and-Place Point (CDPP) model allows the robot to move in the X and Y direction simultaneously. Interestingly, the 'optimal robot movement' can still be preserved even if the robot has to move in the X direction as long as the movement in Y takes longer time than the X movement. This means that the CDPP approach can move the robot in the X direction without adding extra cost to the assembly cycle time, CT. On the contrary, the extended dynamic pick-and-place (EDPP) approach proposed by Hop and Tabucanon (2001b) will pay an extra cost when the robot moves in the X-axis.

Our CDPP approach differs from (Fu and Su, 2000; Hop and Tabucanon, 2001a, b; Su and Fu, 1998; Su et al., 1995; Wang, 1996; Wang et al., 1995, 1997, 1998) as we allow the robot to move in the X and Y direction concurrently whenever necessary, even in the case where the feeder carrier (or PCB table) can move within free movement time. That is, when the feeder carrier (or PCB table) can arrive at the best current pickup (placement) location earlier than the robot arm. Initially, we consider the best current pickup (placement) location as defined by Wang et al. (1998). That is, the pickup (placement) operation, which involves no robot motion in the X-axis. When the feeder carrier is not able to bring the i^{th} component to the i^{th} pickup location, *f'(i)*, or when the PCB table is not able to bring the i^{th} placement point to the i^{th} placement location, *b'(i)*, then robot interception occurs (refer to figure 4.4 and figure 4.5). When robot interception occurs, Fu and Su (2000) allow the robot arm to move at an angle from the Y-axis to catch the feeder carrier or the PCB table at a certain point, while our approach moves the robot arm in X-axis and Y-axis simultaneously and the travelling time is dictated by the maximum of X or Y distance (chebychev distance). If the Y distance is greater than the X distance, the robot arm still performs an 'optimal movement', even though the feeder and/or PCB table are not fast enough to bring the best pickup/placement point on time without the movement of robot arm in the X-axis. Allowing the robot to move in the X direction, even in the case of 'optimal movement', might increase the chances of having many 'optimal movement' for the next placements or pickups operations.

4.6.1 A Chebychev Dynamic Pick-and-Place Point Formulation

Originally, we assumed that the 'optimal pickup' happens if equation (4.2) is true and the 'optimal placement' occurs when equation (4.3) is true. An example of these movements is shown in figure 4.4 and figure 4.5, respectively. In fact, in the CDPP approach, the 'optimal pickup' or 'optimal placement' can still be preserved even though the equation (4.2) or equation (4.3) do not hold as long as the movement of the robot arm in Y takes longer than the movement in X. By default, in optimal feeder carrier movement, this is the case when equation (4.2) is true, the *ith* pickup point is f'(i) (that is f(i) = f'(i)) while $f'(i)_x = b(i-1)_x$. Similarly, in optimal PCB table movement, it is assumed that the *ith* placement point is b'(i) (that is b(i) = b'(i)) and $b'(i)_x = f(i)_x$. This means that in order to test the equation (4.2) or (4.3), we assume the robot only moves in the Y-axis from b(i-1) to f'(i), or from f(i) to b'(i) respectively.

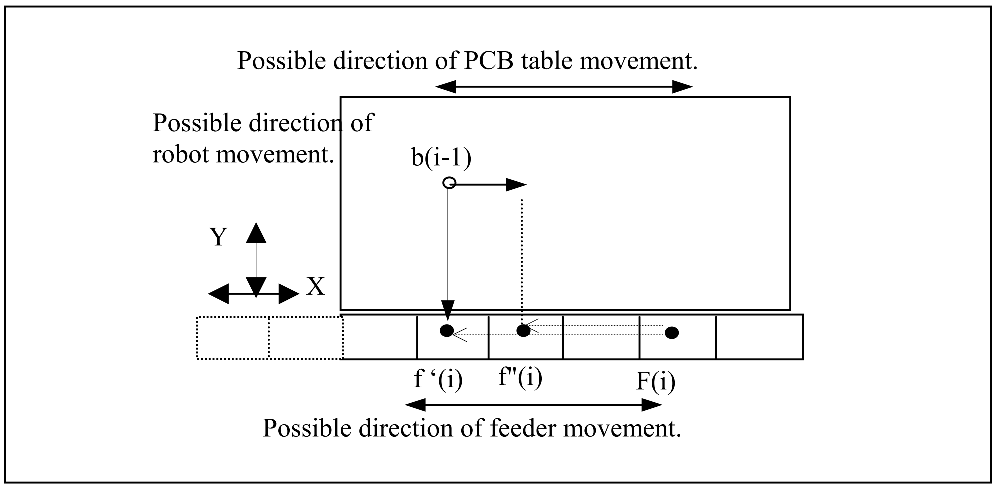

Possible direction of PCB table movement.
Possible direction of robot movement.
b(i-1)
Y
X
f '(i)
f''(i)
F(i)
Possible direction of feeder movement.

Figure 4.4: CDPP model for determining pickup location *f(i)*. The feeder will carry the component from F(i) to f'(i) if robot does not need to move in the X-axis or from F(i) to f''(i) otherwise.

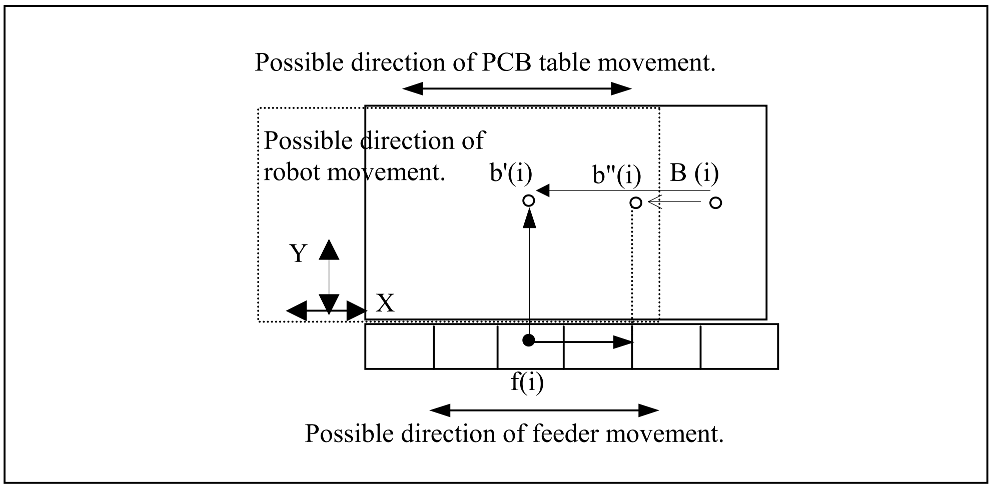

Possible direction of PCB table movement.
Possible direction of robot movement.
b'(i)
b"(i)
B (i)
Y
X
f(i)
Possible direction of feeder movement.

Figure 4.5: CDPP model for determining placement location *b(i)*. The PCB table will move to position the placement point from B(i) to b'(i) if robot does not need to move in the X-axis or from B(i) to b"(i) otherwise.

The detailed calculations of equation (4.2) for the CDPP formulation are:

a) $d_{f(i-1),b(i-1)} = \max(d_{f(i-1)_x,b(i-1)_x}, d_{f(i-1)_y,b(i-1)_y})$

Since the robot can move simultaneously in both the X and Y axis, we use the chebychev distance and calculate the distance between the pickup point *f(i-1)* and the placement point *b(i-1)* as the maximum of the movement in the X and Y direction.

b) $d_{b(i-1),f'(i)} = d_{b(i-1)_y,f'(i)_y}$ where $f'(i)_y = 0$;

Initially, it is assumed that the robot only moves in the Y direction from the placement point b(i-1) to the next pickup point f'(i).

c) $F(i) = F(i)_x = f(i-1)_x + s_{i-1,i}$;

F(i) is a relative feeder slot location (referring to the origin point) containing the *i*th component in the assembly sequence when location *f(i-1)* and *b(i-1)* are already known.

d) $d_{F(i),f'(i)} = d_{F(i)_x,b(i-1)_x} = F(i)_x - b(i-1)_x$;

A distance between the point *F(i)* and the next possible pickup point *f'(i)*. Initially, the next possible pickup is assumed as *f'(i)* that is *f(i)=f'(i)* where $f'(i)_x=b(i-1)_x$. Originally, the feeder carrier should move up to this distance to have an 'optimal movement'. A positive value means the *F(i)* is located at the right side of *b(i-1)*, negative otherwise.

The detailed calculations of equation (4.3) for the CDPP formulation are:

a) $d_{b(i-1),f(i)} = \max\left(d_{b(i-1)_x,f(i)_x}, d_{b(i-1)_y,f(i)_y}\right)$;

Since the robot can move simultaneously in both the *X* and *Y* axis, we use the chebychev distance and calculate the distance between the placement point *b(i-1)* and the pickup point *f(i)* as the maximum of the movement of the robot arm in the *X* and *Y* direction.

b) $d_{f(i),b'(i)} = d_{f(i)_y,b'(i)_y} = d_{f(i)_y,c(i)_y}$ where $f(i)_y = 0$ and $b'(i)_y = c(i)_y$;

Initially, it is assumed that the robot only moves in the Y direction from the pickup point *f(i)* to the next placement point b'(i).

c) $B(i) = B(i)_{x,y}$ where $B(i)_x = b(i-1)_x + c_{(i-1)_x,(i)_x}$ and $B(i)_y = c(i)_y$;

B(i) is a relative coordinate on the PCB (referring to the origin point) where the i^{th} component has to be placed on the PCB during the i^{th} assembly sequence when location *b(i-1)* and *f(i)* are already known.

d) $d_{B(i),b'(i)} = d_{B(i)_x,f(i)_x} = B(i)_x - f(i)_x$;

A distance between the point *B(i)* and the next possible placement point *b'(i)*. Initially, the next possible placement is assumed as *b'(i)* that is *b(i)=b'(i)* where $b'(i)_x = f(i)_x$. Originally, the PCB table should move up to this distance to have an 'optimal movement'.

In order to simplify our terms used in the CDPP formulation we introduce four variables:

a) $p = \frac{d_{f(i-1),b(i-1)}}{V_r} + \theta$ is the total time taken by the robot arm to move from the pickup point *f(i-1)* to the placement point *b(i-1)* together with the time to place the *(i-1)*th component.

b) $q = \frac{d_{b(i-1),f(i)}}{V_r} + \lambda$ is a total time taken by the robot arm to move from the placement point *b(i-1)* to the pickup point *f(i)* together with the time to pickup the *i*th component.

c) $R = pV_f$ is a distance made by the feeder carrier by moving from coordinate *F(i)* to position the *i*th component for the next pickup operation, at *p* time.

d) $Q = qV_b$ is a distance made by the PCB table by moving from coordinate *B(i)* to position the *i*th coordinate on the PCB for the next placement operation, at *q* time.

In optimal pickup (equation (4.2) is true), the CDPP considers two cases (case 1 and case 2) to determine the pickup location, *f(i)*:

Case 1:

*The robot arm moves simultaneously in the X-axis and Y-axis to pick a component at pickup location f(i) (where f(i)=F(i) in this case) and the feeder carrier does not move at all if the X distance between the i*th *and (i-1)*th *PCB coordinate* $c_{i-1,i}$ *is greater than* $d_{F(i),,f'(i)}$*; and the i*th *PCB coordinate is located in the direction in, which the robot arm is moving; and the value of the Y coordinate of the i*th *PCB coordinate* $(c(i)_y)$ *is greater than the absolute value of* $d_{F(i),f'(i)}$.

Then,

$D_x = d_{F(i),f'(i)}$;

$f(i)_x = b(i-1)_x + D_x$;

$Fm(i) = F(i)_x - f(i)_x = [F(i)_x] - [b(i-1)_x + (F(i)_x - b(i-1)_x)] = 0$ *then there is no feeder movement.*

An example of this case 1 is shown in figure 4.6.

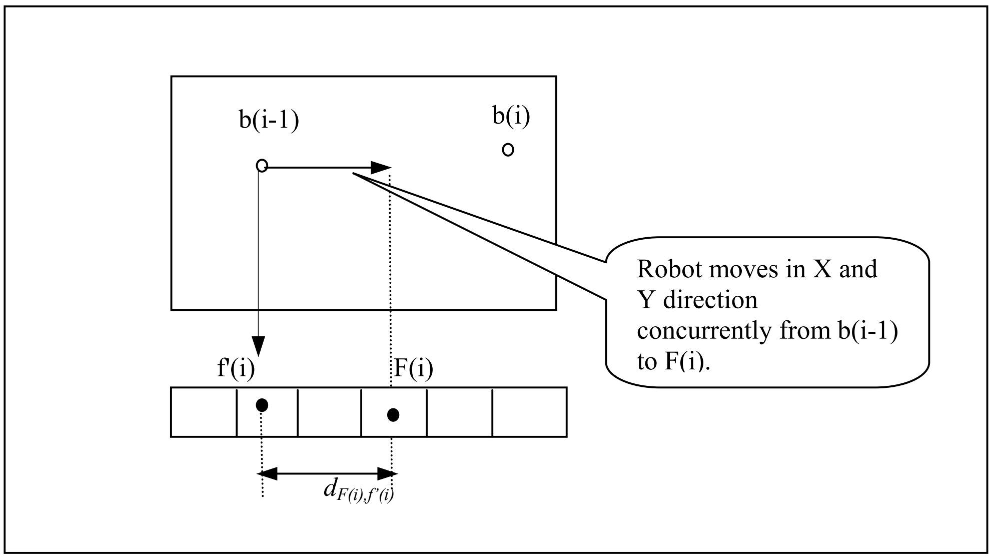

b(i-1)
b(i)
Robot moves in X and Y direction concurrently from b(i-1) to F(i).
f'(i)
F(i)
$d_{F(i),f'(i)}$

Figure 4.6: An example of case 1 where the robot moves in X and Y direction simultaneously from b(i-1) to F(i) to pick the *ith* component whilst the feeder carrier does not move at all.

Case 2:

The robot arm does not move in the X-axis to pick a component at pickup location f(i) if case 1 is not satisfied but equation (2) is true.

Then,

$D_x=0;$

$f(i)_x=b(i-1)_x$ *; (similar to Wang's approach)*

$Fm(i)= F(i)_x-f(i)_x.$

When equation (4.2) is false, then we consider two further cases (case 3 and case 4) to determine pickup point f(i). When equation (4.2) does not hold, the robot arm and the feeder carrier movement time can be expressed by the following equation:

$$\frac{d_{f(i-1),b(i-1)}}{V_r}+\theta+\frac{d_{b(i-1),f''(i)}}{V_r}\geq\frac{d_{F(i),f''(i)}}{V_f} \qquad \textbf{(4.6)}$$

In all conditions for the following cases, the robot arm has to move in the *X*-axis.

Case 3:

Similar to the case 1 except this case considers when equation (4.2) is false. If this case is satisfied, then 'optimal movement' is still preserved even though *the feeder carrier is not fast enough to position the i^{th} component at the pickup location f'(i).*

Case 4:

If case 3 is not satisfied and equation (4.2) is false, then the robot arm moves simultaneously in X and Y direction while the feeder carrier also moves concurrently in X direction to position the i^{th} component to the new relative pickup location. The robot arm stops moving in X when it meets the feeder carrier at $f''(i)_x$.

Then;

$$D_x = \frac{[\text{abs}(d_{F(i),f'(i)}) - R] * Vr}{Vr + Vf};$$

$f(i)_x=b(i-1)_x + D_x$ *if* $s_{i-1,i}$ *is positive, or*

$f(i)_x=b(i-1)_x - D_x$ *if* $s_{i-1,i}$ *is negative ;*

$Fm(i)=F(i)_x-f(i)_x;$

In case 4, the 'optimal movement' can still be preserved if the absolute value of D_x is less than the value of Y coordinate of the *(i-1)*th PCB coordinate *(c(i-1))*.

Similarly, to determine the placement location *b(i)*, we will consider two cases (case 5 and 6) when equation (4.3) is true (optimal movement):

Case 5:

The robot moves simultaneously in the X-axis and Y-axis to place a component at placement location b(i) and the PCB table does not move at all if the distance between the feeder slots for the i^{th} and $(i+1)^{th}$ components is greater than $d_{B(i),b'(i)}$, and the feeder slot containing the $(i+1)^{th}$ component is located in the direction in which the robot arm is moving; and the value of the Y coordinate of the i^{th} PCB coordinate ($c(i)_y$) is greater than the absolute value of $d_{B(i),b'(i)}$.

Then,

$D_x=d_{B(i),b'(i)},;$

$b(i)_x=f(i)_x + D_x;$

$Tm(i)=B(i)_x-b(i)_x=0$ *then there is no PCB table movement.*

Case 6:

The robot only moves in the Y-axis to place a component at placement location b(i) if equation (4.3) is true but case 5 is not satisfied.

Then,

$D_x=0;$

$b(i)_x=f(i)_x$; *(similar to Wang's approach)*

$Tm(i)= B(i)_x-b(i)_x$.

If equation (4.3) is false, then we consider another two cases (case 7 and 8) to determine the placement location *b(i)*. When equation (4.3) does not hold, the robot arm and PCB table movement time can be expressed by the following equation:

$$\frac{d_{b(i-1),f(i)}}{V_r}+\lambda+\frac{d_{f(i),b''(i)}}{V_r}\geq\frac{d_{B(i),b''(i)}}{V_b} \qquad \textbf{(4.7)}$$

In all conditions for these cases, the robot has to move in the X direction.

Case 7:

Similar to the case 5 except this case is considered when equation (4.3) is false and equation (4.7) is true. If this case is satisfied, then 'optimal movement' is preserved even though the PCB table is not fast enough to position the ith *PCB coordinate at placement point b'(i).*

Case 8:

If case 7 is not satisfied, equation (4.3) is false and equation (4.7) is true, then the robot arm moves simultaneously in the X and Y direction while the PCB table also moves concurrently in the X direction to position the ith *PCB coordinate at the new relative placement position. The robot arm stops moving in X when it meets the placement location b"(i)*$_x$.

Then;

$$D_x=\frac{\left(abs(d_{B(i),b'(i)})-Q\right)*V_r}{V_r+V_b};$$

$b(i)_x=f(i)_x + D_x$ *if* $c_{i-1,i}$ *is positive, or*

$b(i)_x=f(i)_x - D_x$ *if* $c_{i-1,i}$ *is negative ;*

$Bm(i)=B(i)_x-b(i)_x;$

Again, in case 8, the 'optimal movement' can still be preserved if the absolute value of D_x is less than the value of Y coordinate of the i^{th} PCB coordinate *(c(i))*.

4.6.2 Methodology for Component Pick-and-Place Sequencing

Since our work is focusing upon improving the feeder setup, we follow the method used in Wang et al. (1998) in determining the component placement sequence. This allows us to make a fair comparison with Wang's approach in our experiments. To schedule the component placement sequence, the placement points are sequenced from left to right starting with the smallest X at the left lowermost corner of the PCB then with larger Y if more than one coordinate has the same value of X.

4.6.3 A Chebychev Dynamic Pick-and-Place Point Robot Motion Control

In this section, we investigated the effectiveness of a Chebychev Dynamic Pick-and-Place Point (CDPP) robot motion control. Therefore, we used Wang's approach (Wang et al., 1997) for feeder setup for the aide of comparison. To decide the feeder setup, components are assigned to a specific feeder slot such that the total exchange frequency of all adjacent slot pairs has the maximum value. The exchange frequency is an index that counts the exchange frequency between two different component types for succeeding pickups.

The feeder setup problem is converted to a travelling salesman problem (TSP) by associating a feeder slot as a node (or city) and the exchange frequency as the arc (or distance) connecting the two nodes (or cities). The algorithm of the feeder setup is (adopted from Wang et al., 1997):

- Generate a K by K matrix of exchange frequency for each pair of component types based on the previously obtained component placement sequence.

- Symmetrically add, $F_{gz} + F_{zg}$, where the exchange frequency between component type *g* and type *z* is fixed regardless of whether the pickup order is from component type *g* to type *z* or vice versa.
- Subtract from a large number (a number larger than all values in the matrix) in order to convert the feeder setup problem to a travelling salesman problem such that the aim is to find the shortest path.
- Assign components to feeder slots by applying any heuristic that can be applied to the travelling salesman problem.

In this section, we only use a constructive heuristic to arrange the feeder slots since our work only focuses on the robot motion control specification. However, we believe that by applying an even better heuristics in the feeder setup and component placement sequencing, we can gain even better assembly cycle times by reducing the feeder and PCB table movements.

4.6.3.1 Testing and Results

In our experiments we assume that the feeder carrier and PCB table are positioned close to each other in order to minimise the robot arm travel distance (Ahmadi and Mamer, 1999). We also assume that all components use the same nozzle; and the speed of robot arm, PCB table and feeder carrier, are fixed for all components. Then, we further assume that the components are assigned to the placement machine and all components are the same size. The pick up and placement time are set as 0.5 unit time and the size of each feeder slot is 4 unit lengths.

The placement points are generated randomly. Components are assigned to a specific feeder slot such that the total exchange frequency of all adjacent slot pairs has the maximum value. We apply the seven factors (table 4.1) of parameters as used by Su et al. (1995) and Hop and Tabucanon (2001b). To demonstrate the performance of our approach, we choose the length of the PCB, the width of the PCB, the speed of robot arm, the speed of feeder carrier, the speed of PCB table as 40, 15, 12, 2.5 and 3 respectively (as shown in table 4.1). The assembly points are chosen as 50 or 100 whilst the number of component types as 5, 10, 20, 30 or 40 (also shown in table 4.1).

TABLE 4.1: THE SEVEN FACTORS OF EXPERIMENTAL PARAMETERS

Factors	**Levels (low/high)**
Number of assembly points (N)	50/100
Number of component types (K)	5/10/20/30/40
Length of PCB (BL)	40 (unit distance)
Width of PCB (BW)	15 (unit distance)
Speed of robot (Vr)	12 (unit distance/unit time)
Speed of feeder carrier (Vf)	2.5 (unit distance/unit time)
Speed of PCB table (Vb)	3 (unit distance/unit time)

For each combination of N and K, we performed five runs with the same feeder setup. Since this is a deterministic approach (i.e. each run will produce the same result), we need to use a different dataset (which are randomly generated) for each run. However, the number of component types, K is fixed for each combination of N and K such that we can use the same feeder setup for the five runs.

The computational results are summarised in table 4.2 and are averaged over five runs. The ‘number of optimal movement’ column in table 4.2 stated the average of the ‘optimal movement’ obtained from five runs. ‘Optimal movement’ occurs when the feeder carrier (or PCB table) can move within free movement time. That is, when the feeder carrier (or PCB table) can arrive at the best current pickup (placement) location earlier than the robot arm.

The results show that our approach is superior to Wang’s (Wang et al., 1998) in all tests. Our approach performs, an average, 3.29% better than Wang’s when considering assembly cycle time, 55.54% improvement of ‘optimal movement’, shorter feeder movement distance (10.21% improvement) and shorter PCB movement distance (19.12% improvement) compared to Wang’s approach (Wang et al., 1998).

Figure 4.7, 4.8, 4.9 and 4.10 show the results of each run comparing the CDPP approach with that of Wang et al. (1998).

TABLE 4.2: THE AVERAGE RESULTS OF FIVE RUNS

Combination		**CT (assembly cycle time)**			**Number of optimal movement, ϑ**			**Feeder movement distance, Λ**			**PCB table movement distance, ξ**		
N	K	CDPP	WA	I (%)	CDPP	WA	$^{+}$I (%)	CDPP	WA	I (%)	CDPP	WA	I (%)
50	5	117.25	119.28	1.73	88.00	62.80	40.13	148.18	168.30	13.58	106.85	143.79	34.57
50	10	127.57	132.40	3.79	72.20	46.80	54.27	201.85	227.87	12.89	164.06	212.13	29.30
50	20	151.96	158.73	4.46	55.20	35.80	54.19	286.96	317.19	10.54	285.39	332.50	16.51
50	30	168.00	173.42	3.22	56.40	35.60	58.43	364.91	396.98	8.79	337.29	372.43	10.42
50	40	167.06	170.67	2.16	66.60	48.40	37.60	403.95	436.15	7.97	297.37	324.16	9.01
100	5	230.95	234.66	1.61	174.80	129.00	35.50	288.63	333.65	15.60	174.43	231.50	32.72
100	10	248.03	258.40	4.18	143.00	87.20	63.99	406.66	456.59	12.28	317.28	411.27	29.62
100	20	323.93	339.00	4.65	102.60	58.40	75.68	635.51	687.07	8.11	630.70	713.15	13.07
100	30	384.66	400.86	4.21	85.60	48.40	76.86	789.21	852.58	8.03	846.73	931.13	9.97
100	40	465.12	478.65	2.91	80.00	50.40	58.73	1019.92	1064.24	4.35	1104.91	1171.19	6.00
Average:				**3.29**			**55.54**			**10.21**			**19.12**

*Note: WA=Wang's approach; I =Improvement over Wang's approach (I % = (WA-CDPP)*100/CDPP)); $^{+}$I = (CDPP- WA)*100/WA))*

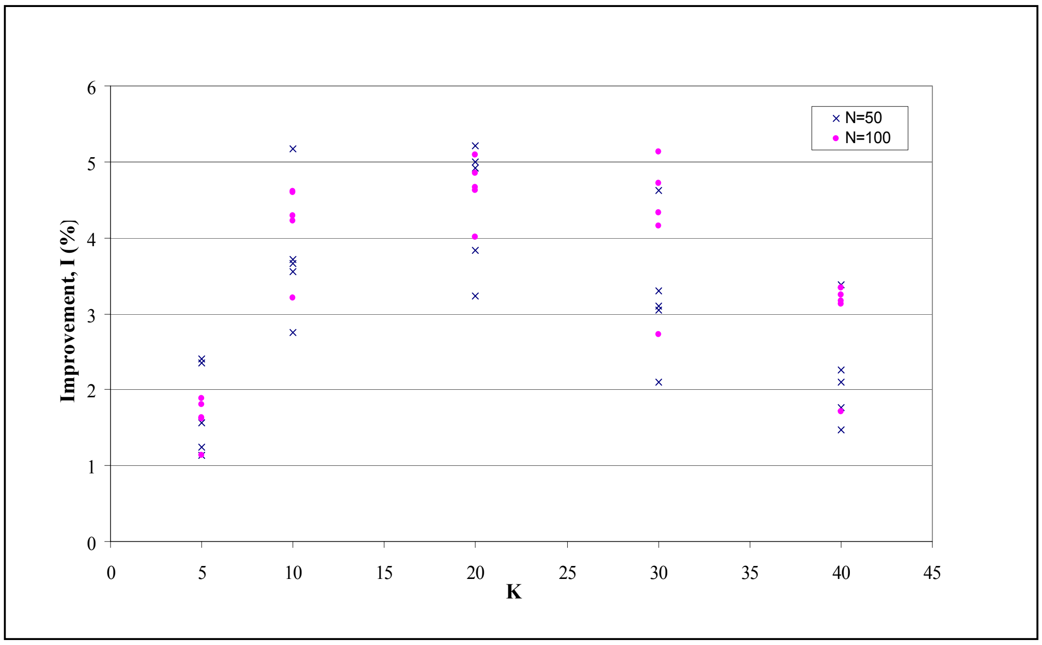


Figure 4.7: **A comparison between CDPP and Wang et al. (1998) in terms of assembly cycle time, CT where I = $(CT_{WA} - CT_{CDPP})*100/ CT_{CDPP}$.**

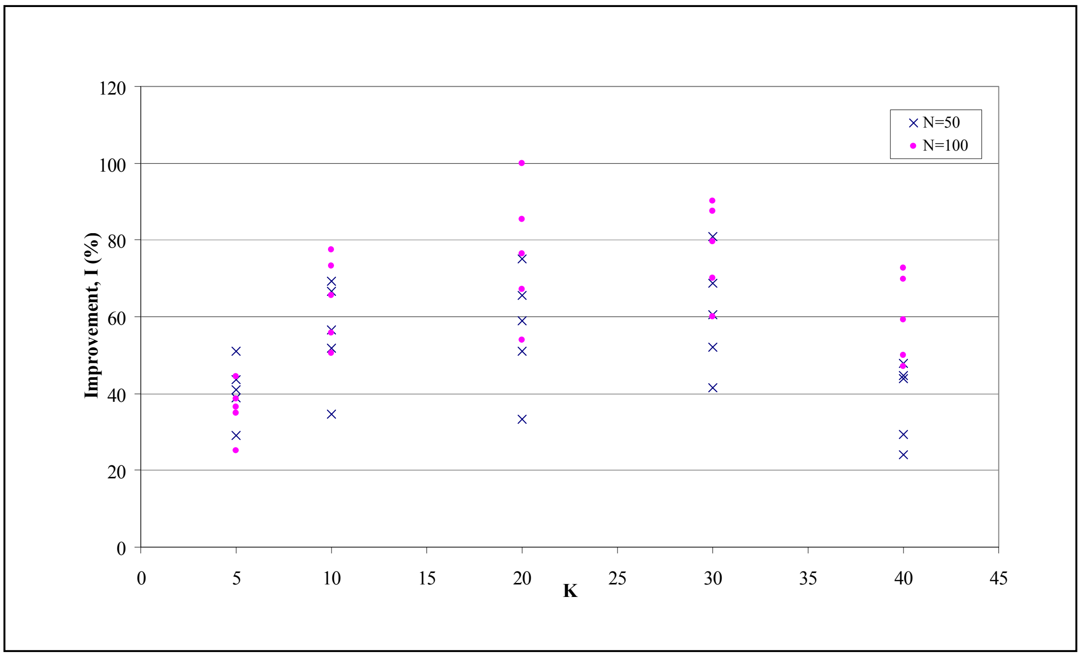


Figure 4.8: **A comparison between CDPP and Wang et al. (1998) in terms of number of 'optimal movement', ϑ where I=$(\vartheta_{CDPP} - \vartheta_{WA})*100/\vartheta_{WA}$.**

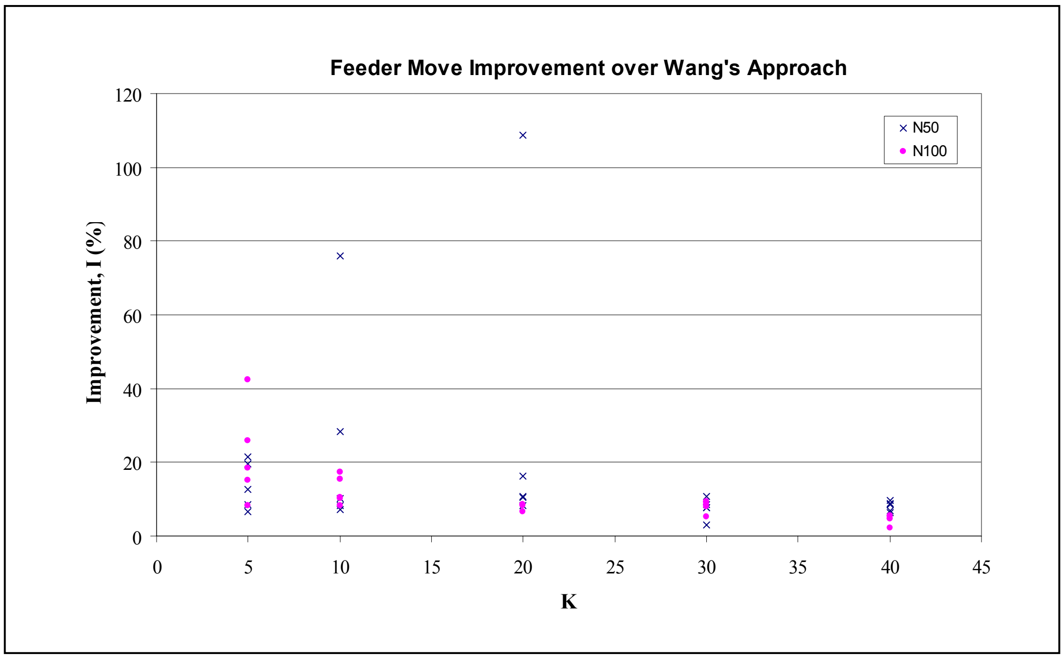


Figure 4.9: A comparison between CDPP and Wang et al. (1998) in terms of feeder movement distance, Λ where $I=(\Lambda_{WA} - \Lambda_{CDPP})*100/\Lambda_{CDPP}$.

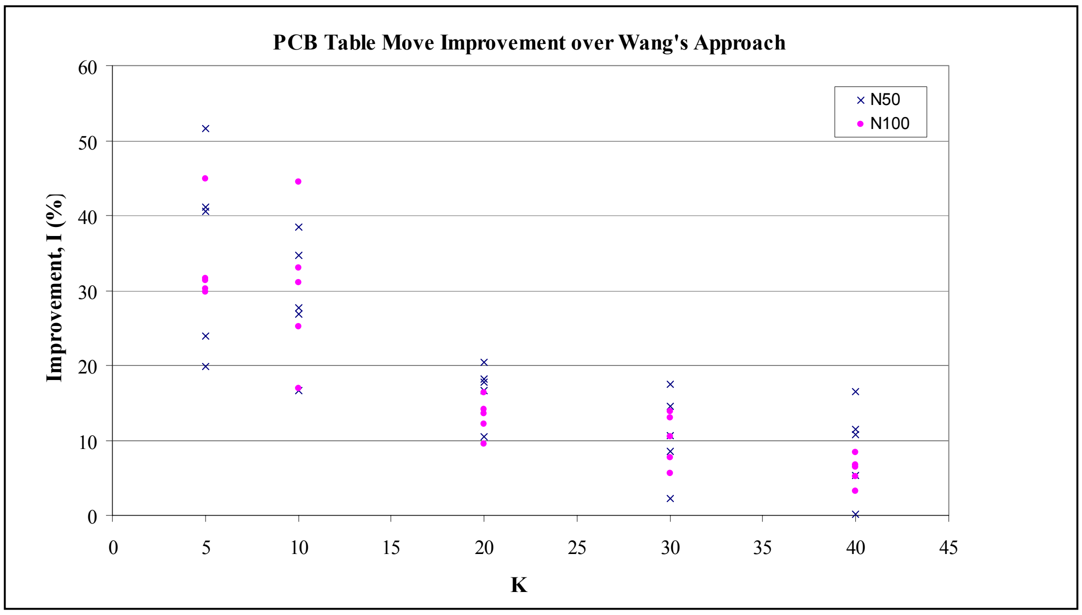


Figure 4.10: A comparison between CDPP and Wang et al. (1998) in terms of PCB table movement distance, ξ where $I=(\xi_{WA} - \xi_{CDPP})*100/\xi_{CDPP}$.

From figure 4.7 and 4.8, the CDPP approach shows a greater improvement over Wang's approach when *K=20* and *K=30* whilst a smaller improvement when *K=5* and *K=40*. This might be due to the size of the PCB, which is fixed across all tests. The total length of feeder slots occupied by 20 or 30 component types (i.e. more than double that of the PCB length), might be the 'ideal' case for the CDPP to outperform the Wang's approach. For these cases (i.e. when K=20 or K=30 with the other fixed experimental parameters as shown in table 4.1), Wang's strategy, although allowing movements in both the X and Y directions can only generate optimal movement when the arm moves in the Y direction only. The CDPP approach, proposed here, is able to generate optimal movements even if the robot arm moves in both X and Y directions. This optimal movement occurs when the movement in the X direction is less than the movement required the Y direction. This results in the CDPP approach improving the assembly cycle time, *CT*. However, when the number of component type is smaller (*K=5*) or very big (*K=40*), Wang's approach performs as well as the CDPP approach. Wang's approach might be able to produce many 'optimal movement' when *K=5* (with the other fixed experimental parameters as shows in table 4.1) and this reduces the performance (number of 'optimal movement' and *CT*) gap between the CDPP and Wang's approaches. On the contrary, when *K=40*, the total length of the feeder slots occupied by 40 component types is too large, which causes a difficulty for both approaches to have many 'optimal movement' and this argument is supported by the results in figures 4.9 and 4.10 where both approaches have almost an equal amount of feeder carrier and PCB table movement.

Results in figure 4.9 and 4.10 show that the performance of CDPP (in terms of feeder and table move distance improvement) and when compared to Wang's approach, reduces as the number of component types, *K*, increases. This might be due to the CDPP approach allowing the robot arm to move in the X direction and still generate an optimal movement (i.e. when the required X movement is less than the required Y movement). This appears to help reduce the PCB table and feeder carrier movement for a smaller number of component types (*K*=*5*).

However, for a larger value of *K*, this strategy might not help, which results in almost an equal performance for both approaches in term of reducing the PCB table and feeder carrier movements.

4.6.3.2 Discussion

The Chebychev dynamic pick-and-place (CDPP) motion control eliminated the unnecessary movement by looking forward to the next PCB coordinate when determining the current pickup location and looking forward the next feeder slot when determining the current placement location. The CDPP formulations are constructed based on the aims of minimising robot assembly time, feeder movements and PCB table movements. The main difference between our CDPP model and the previous DPP (and EDPP) was that our CDPP calculated the robot arm movement distance as the maximum of the movement in Y or the movement in X (a chebychev distance) since our robot arm can move in X-axis and Y-axis concurrently, whilst the previous DPP (and EDPP) calculated the robot arm movement as a euclidean distance. This work has shown an improvement compared to Wang's DPP approach. For all tests, our CDPP approach is superior to Wang's approach (i.e. all '*I*' have positive values) when considering assembly cycle time, 'number of optimal movement', shorter feeder movement distance and shorter PCB movement distance. Therefore, we further explored the CDPP approach by introducing a triple objective function to improve the feeder setup in order to gain even better results.

4.6.4 A Triple Objective Function

In this section we extend the work in section 4.6.3 by introducing a triple objective function with a CDPP approach to optimise the sequential pick-and-place SMD placement machine. In section 4.6.3 we focused on improving the robot motion control, whereas this section addresses improving the feeder setup. The aims are to minimise the robot assembly time, the feeder movements and the PCB table movements.

Wang et al. (1998) developed a heuristic for feeder setup by placing components in feeder slots such that all adjacent slots have maximum exchange frequency pickups involving multiples of the same component type. Hence, they arrange the feeder slots by searching for the maximum total exchange frequency. By doing this, Wang et al. (1998) assumed that maximising the total exchange frequency can lead to the minimum assembly cycle time (CT).

However, Hop and Tabucanon (2001a) argued that assigning component feeders to slots based on maximum exchange frequency of the component types, may increase the feeder rack movement if components are not assembled by their type. Contrarily, our experimental results (refer to table 4.4) show that Wang's approach for feeder setup can gain better (i.e. less) feeder movement. However, we also found that maximising (actually minimising after converting to a TSP problem) the total exchange frequency does not necessarily lead to the minimum CT. To show this, we ran twenty tests on a small dataset using full enumeration to determine the optimum feeder setup. The placement sequences are generated based on Wang's approach. We defined three objective functions, which are to maximise the exchange frequency, minimise the CT and minimise the triple objective function. The triple objective function aims to minimise the CT whilst also minimising the feeder and PCB table movements. However, the main objective remains to minimise the CT but it would be beneficial if we can also minimise the feeder and PCB table movements. Reducing these movements may prolong the life cycle of placement machine even if it does not affect the throughput rate of the machine. By running a few preliminary tests we find that there is not much difference in CT among different feeder setups but a lot of variance in PCB table and feeder movements. If the triple objective function is determined without normalisation, then the output will be biased to have a better feeder movement and PCB table movement rather than better CT. Hence we normalise as follows:

Changes in *CT*,

$$\Delta C = \frac{newC - oldC}{newC}$$

Changes in PCB table movement,

$$\Delta P = \frac{newP - oldP}{newP}$$

Changes in feeder movement,

$$\Delta M = \frac{newM - oldM}{newM}$$

where *C, P* and *M* are referring to *CT*, PCB table movement and feeder movement respectively. *ΔC* is calculated as a difference of *CT* between the current solution (newCT) and the best solution (oldC) previously obtained. Likewise, *ΔP* and *ΔM* are calculated in a similar way. A negative value means that the new solution is better than the best solution previously obtained. To introduce more flexibility into our triple objective function, we incorporate the changes of *CT*, PCB table movement and feeder movement with weighted parameters. The weighted parameters will determine the importance of the *CT*, PCB table movement and feeder movement. Therefore, the triple objective function is:

$$F = \min\left(W_c * \Delta C + W_p * \Delta P + W_m * \Delta M\right) \tag{4.8}$$

where W_c, W_p and W_m are the weighted parameters for *CT*, PCB table movement and feeder movement respectively. By adapting the weighted parameters, we can gain better *CT* without ignoring the feeder movement and PCB table movement. Of course, if one assumes that *CT* is the most important factor, then the W_c parameter can be set to a suitably large value to ensure that the algorithm will search for a better *CT* and then when there are more than one solution having the same value of *CT*, the algorithm will select the better PCB table movement and feeder movement.

In equation 4.8, we are looking for the minimum value of *F*. *F* is calculated as the sum of changes in the *CT*, PCB table movement and feeder carrier movement and the multiplication of their weighted parameters. The comparison is made with the best solution that was previously obtained in the enumeration. A positive value of *F* means that the current solution is worse and zero value indicates that the current solution is similar to the best solution obtained in terms of the *F* value. However, a negative value of *F* shows that the current solution is superior to the best solution been obtained. If a negative value of *F* is obtained, then the best solution is updated with the new solution.

4.6.4.1 Testing and Results

To decide the feeder setup, we use three approaches:

a) In minimising (actually maximising) the exchange frequency, component feeders are assigned to a specific feeder slot such that the total exchange frequency of all adjacent slot pairs has the maximum value (as in Wang et al., 1997). In this approach we use a constructive heuristic (nearest neighbourhood) to assign components to feeder slots by choosing the first component type to be placed onto the PCB and assigning it to slot 0 (as an initial city in the TSP problem). Then, the rest of the component feeders are assigned to the slots using the nearest neighbourhood heuristic to construct an initial feeder setup. Starting from initial feeder setup, we run a full enumeration with the aim of minimising (actually maximising) the total exchange frequency of all adjacent slot pairs. In every iteration, the new solution is compared to the best obtained solution, in this case, the solution, which has the minimum exchange frequency.

b) In minimising the assembly cycle time (*CT*), component feeders are assigned to a specific feeder slot such that the total *CT* is minimised. To simplify our work, we use the same constructive heuristic as in the first approach to generate an initial solution. Instead of searching for a minimum (actually maximum) exchange frequency, we directly search for the minimum *CT*. Again, we run a full enumeration to obtain an optimal *CT* (shown in table 4.4; *CT* under '*Minimise CT*'). In each iteration, the *CT* of the current solution is compared to the best obtained *CT*.

c) Finally, in minimising the triple objective function (*F*), component feeders are arranged in the feeder slots such that the *CT* and the movement of PCB table and feeder carrier are minimised. Again, we use the same constructive heuristic to obtain the initial feeder setup. A full enumeration is performed to find the global optimal solution that has the minimum *CT* with a better movement of feeder carrier and PCB table. The optimal solution is dependent on the chosen value of weighted parameters (W_c, W_p *and* W_m). If W_c is a suitably large value, then the optimal solution will be the optimal *CT* but if there exist solutions with the same *CT* value, then the solution, which has better feeder carrier and PCB table moves will be chosen.

The assumptions of this experiment are the same as in section 4.6.3.1. In our experiments, we choose W_c=20, W_p=1 and W_f=1. This proportion is suitably large enough to emphasise the importance of assembly cycle time, CT. The placement points are generated randomly. We apply the seven factors (table 4.3) of parameters as used in (Su et al., 1995; Hop and Tabucanon, 2001b). The pick up and placement time are set as 0.5 unit time and the size of each feeder slot is 4 unit lengths. For the purpose of generating the random placement points, we choose the length and the width of the PCB as 40 and 10 unit length, respectively, such that the random placement points fall within these limits. The assembly points are chosen as 50 whilst the number of component types is 8 (as shown in table 4.3). For simulating the assembly cycle time, feeder carrier movement and PCB table movement, we set the speed of the robot arm, feeder carrier and PCB table as 6, 5 and 4 unit distance/unit time respectively (as shown in table 4.3). We use a different experimental parameter values to show the robustness of the CDPP approach that can work across any dataset. The computational results are shown in table 4.4 and are obtained from twenty runs (we do twenty runs here, as opposed to the five runs earlier as the early work was attempting to demonstrate that the technique worked). New datasets are randomly generated in each test (i.e. we use different problem instances for each test). The CDPP approach is used to determine the pickup-and-placement location during the component assembly process. The *CT*, feeder movement (*FM*) and PCB table movement (*PM*) are calculated based on the CDPP approach.

TABLE 4.3: THE SEVEN FACTORS OF EXPERIMENTAL PARAMETERS

Factors	**Levels (low/high)**
Number of assembly points (N)	50
Number of component types (K)	8
Length of PCB (BL)	40 (unit distance)
Width of PCB (BW)	10 (unit distance)
Speed of robot (Vr)	6 (unit distance/unit time)
Speed of feeder carrier (Vf)	5 (unit distance/unit time)
Speed of PCB table (Vb)	4 (unit distance/unit time)

As we are dealing with a minimisation problem, smaller values are better (referring to table 4.4, figure 4.11, figure 4.12, figure 4.13 and figure 4.14 and figure 4.15). The results shown in table 4.4, figure 4.11, figure 4.12 and 4.13 clearly indicate that there is no relation between the exchange frequency and the *CT*. Figure 4.11 shows that in all tests the minimum exchange frequency does not coincide with the minimum *CT*. In test 5 (refer table 4.4), for example, the minimum exchange frequency is 22 with the *CT* being 147.07 unit time, whilst the minimum *CT* is 142.19 unit time obtained when the exchange frequency is 31. Indeed, figure 4.13 clearly shows that for all tests, the *CT's* of the minimising exchange frequency approach (*Minimise EF*) are the worst compared to the other two approaches. Of course, the *CT's* of minimising *CT* is the best (for all tests) since it searches for the optimum *CT* (as we performed a full enumeration for all tests). We suspect that we cannot reach the optimal *CT* if we are just looking for the minimum exchange frequency. Hence, assigning component feeders to slots based on exchange frequency can be considered as an 'incorrect' strategy. These results are inconsistent with Wang et al. (1998), where they claim that minimising (actually maximising) the total exchange frequency can lead to the minimum *CT*. As such, they assigned component feeders to slots so that the total exchange frequency of all adjacent slot pairs has the minimum value (or actually maximum) assuming they would obtain the minimum *CT*.

TABLE 4.4: A COMPARISON BETWEEN THREE DIFFERENT OBJECTIVE FUNCTIONS THAT ARE MINIMISING THE CT, MINIMISING THE EXCHANGE FREQUENCY AND MINIMISING THE TRIPLE OBJECTIVE FUNCTION.

	Minimise CT				**Minimise Exchange Frequency**				**Minimise F**			
Test	**EF**	**CT***	**FM**	**PM**	**EF***	**CT**	**FM**	**PM**	**EF**	**CT**	**FM**	**PM**
1	27	135.25	344.57	74.79	19	137.48	323.77	101.26	27	135.27	344.57	74.79
2	20	147.70	352.66	80.20	16	150.68	336.70	104.29	20	148.32	365.98	67.69
3	17	138.82	304.39	110.82	12	143.71	280.40	152.95	15	139.65	304.09	69.49
4	25	144.21	273.88	71.50	19	145.28	237.09	76.77	25	144.21	273.88	71.50
5	**31**	**142.19**	**317.89**	**95.68**	**22**	**147.07**	**291.05**	**119.95**	**29**	**143.53**	**317.91**	**65.32**
6	31	124.55	342.83	129.79	21	127.75	294.91	128.91	26	124.76	323.98	105.39
7	27	143.47	344.73	64.93	21	146.43	302.45	119.66	22	144.47	318.52	58.61
8	29	139.29	317.93	126.00	24	141.37	265.30	94.86	28	140.50	296.89	72.48
9	35	136.17	342.86	95.89	28	138.76	332.24	117.47	32	136.24	371.68	71.30
10	17	137.29	384.30	82.64	11	141.52	332.57	110.52	17	137.29	384.30	82.64
11	32	140.69	387.59	74.41	23	141.73	347.55	83.94	32	140.76	352.09	54.98
12	29	135.39	377.29	133.55	20	139.05	333.45	121.45	28	136.61	371.77	84.05
13	22	133.51	305.55	102.46	17	136.78	303.77	118.28	23	134.66	302.14	73.11
14	20	146.40	332.45	75.75	15	149.42	330.36	118.13	20	146.80	371.89	56.18
15	26	132.99	351.39	90.33	19	133.93	297.75	90.35	19	133.93	297.75	90.35
16	26	149.81	347.68	83.00	18	151.76	291.68	111.45	24	150.73	350.18	51.45
17	24	137.32	316.50	82.93	22	141.45	345.57	96.18	25	137.77	333.61	63.48
18	25	137.45	322.00	99.98	20	141.56	317.82	122.35	26	137.87	349.80	59.83
19	26	139.76	350.82	96.20	17	141.58	273.80	105.20	31	140.03	365.09	52.51
20	22	138.14	332.80	104.19	15	140.11	299.11	110.64	19	138.59	314.89	95.03
Ave	**26**	**139.02**	**337.51**	**93.70**	**19**	**141.42**	**306.87**	**110.23**	**24**	**139.60**	**335.55**	**71.01**

Note: EF = Exchange frequency; CT = Assembly cycle time FM = The distance of Feeder movement
PM = The distance of PCB table movement; CT = Optimum assembly cycle time*
EF = Optimum exchange frequency.*

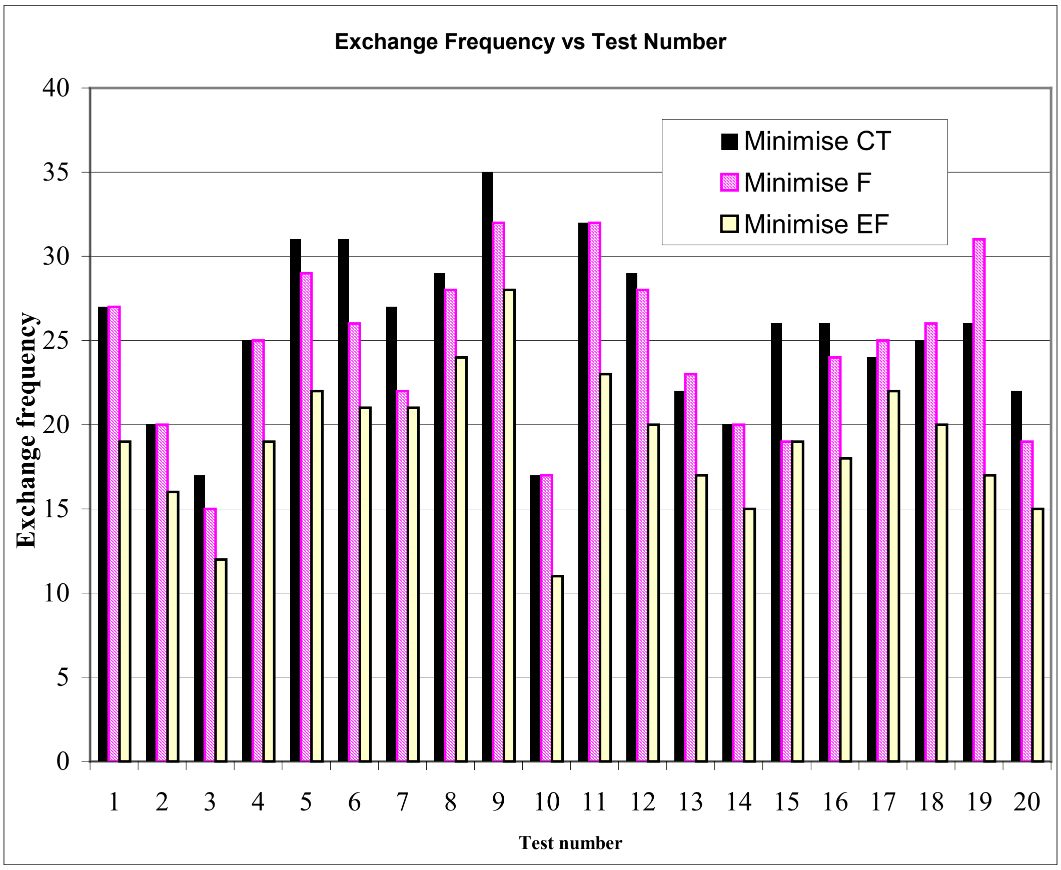


Figure 4.11: A comparison among three objective functions based on the exchange frequency.

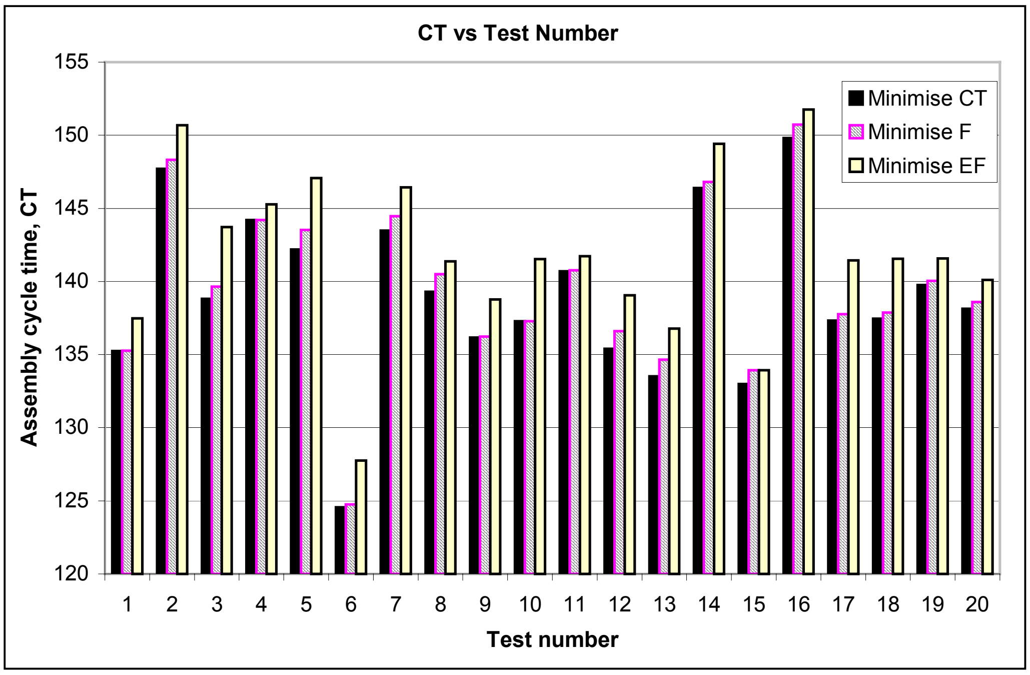


Figure 4.12: A comparison among three objective function based on the CT (bar chart).

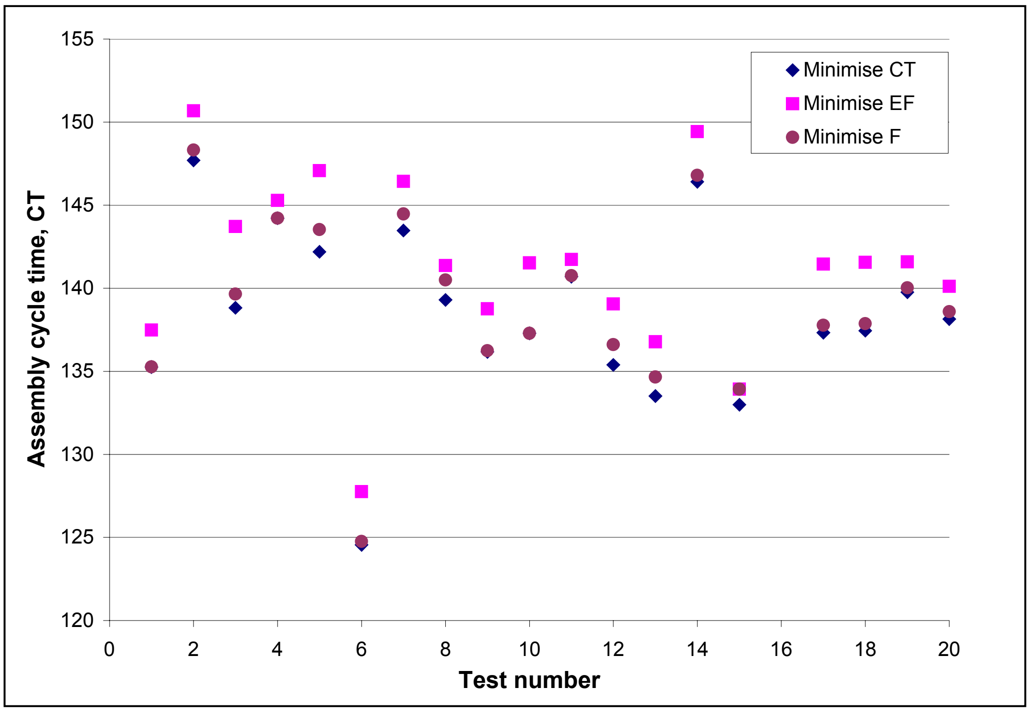


Figure 4.13: A comparison among three objective function based on the CT (line chart).

The results also indicate that we might obtain 'near optimal' *CT* if we use the triple objective function, *F*, in searching for the optimal solution (i.e. the *CT's* values obtained by minimising the triple objective function, *F* approach is fairly close to the *CT's* values obtained by minimising the *CT* approach). This is clearly shown in figure 4.12 and 4.13, where the triple objective function, *F* obtained an 'optimal' *CT* in 25% of the tests (over 20 test) whilst the other 75% obtained 'near optimal' *CT*. However, the strategy of minimising the exchange frequency is unable to obtain any optimal *CT* over 20 tests. In addition, by using the triple objective function, we can gain better movement of the feeder carrier and PCB table compared to the solution obtained by minimising *CT* (refer to figure 4.14 and figure 4.15). For example, in test 5 (table 4.4), by minimising the *CT*, we obtain 142.19 unit time as the minimum *CT* (the optimal *CT*) with the movement distances of feeder carrier and PCB table being 317.89 and 95.68 unit distance, respectively. However, by using the *F* objective function, we obtain 143.53 unit time, 317.91 and 65.32 unit distance for the movement distance of the feeder carrier and PCB table, respectively (referring to test 5 in table 4.4).

From table 4.4 and figure 4.14 we also can see that in all tests the strategy of minimising the exchange frequency can provide a better feeder carrier movement compared to the other two strategies. Hence, we can conclude that assigning component feeders to slots based on minimising (actually maximising) the total exchange frequency of all adjacent slot pairs can provide better feeder carrier movement. Unfortunately, this result contradicts with Hop and Tabucanon (2001a) where they claimed (although show no evidence) that when the feeder assignment is arranged based on maximising the exchange frequency of component types, it might increase the feeder carrier movement if components are not assembled by their type. In fact, our approach does not assemble components based on their type. Of course, when the components are assembled based on component type, the movement of the feeder carrier can be eliminated but we may pay an extra cost in the robot arm movement, which may increase the *CT*. Since the robot arm movement is considered as 'unavoidable', and the movement of the feeder carrier and PCB table can be ignored if they can move within the 'free' movement time (that is the robot arm does not have to wait for them), assembling components based on their type may not be a good strategy as it may increase *CT*.

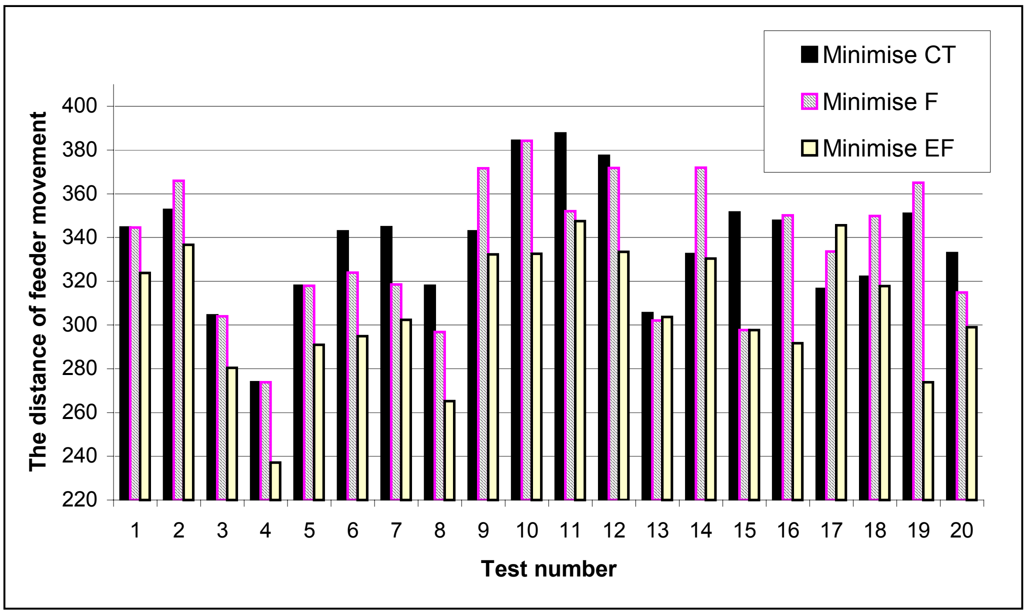


Figure 4.14: A comparison among three objective function based on the distance of feeder movement.

In figure 4.15, we can observe that by minimising *F*, we obtained better

PCB table movement in 80% of the tests. In fact, we might achieve an optimal *CT* with a good movement of PCB table and feeder carrier when the weighted parameter of *CT, Wc*, is set to a suitably large value. On the contrary, we can only achieve the optimal *CT*, but not always minimal PCB table and feeder movement when we only attempt to minimise *CT*. Sometimes, when there exist solutions with the same *CT*, we need other factors to determine which solution should be chosen. In this case, our triple objective function provides an advantage in choosing a better solution quality.

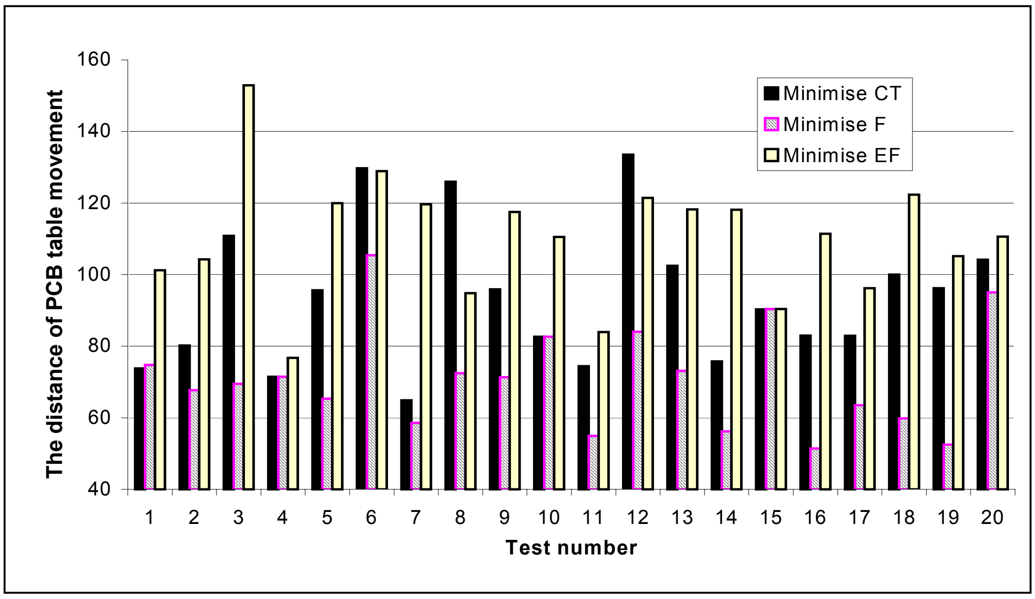


Figure 4.15: A comparison among three objective function based on the distance of PCB table movement.

4.6.4.2 Discussion

In this work we have developed three strategies for feeder setup that assigns component feeders to slots by minimising the assembly cycle time (*CT*), minimising the total exchange frequency and minimising the triple objective function. We found that minimising (actually maximising) the total exchange frequency can only provide better feeder carrier movement rather than a better *CT*.

We have introduced the triple objective function with a CDPP approach in our formulation to improve the feeder setup. The function aims to minimise the

CT together whilst also minimising the feeder carrier and PCB table movements. By using this strategy we might find a better solution quality with a better movement of feeder carrier and PCB table. In summary, minimising the triple objective function strategy might provide better solution quality compared to the strategies of minimising the *CT* and minimising the exchange frequency.

4.7 Summary

In the DPP model, the pickup-and-placement points were dynamically changed based on the movement of the robot arm, the PCB table and the feeder carrier where all of them can vary their speed. Su et al. (1995) argue that the DPP was superior to FPP. However, we found that the Wang's DPP model only considered the current movement. Wang's DPP model tried to maintain the fixed pickup-and-placement location as much as possible unless this leads to robot idling. Hence, the DPP model may still have unnecessary movement. Thus, in our CDPP, we eliminated the unnecessary movement by looking forward to the next PCB coordinate when determining the current pickup location and looking forward the next feeder slot when determining the current placement location.

The CDPP formulations are constructed based on the aims of minimising robot assembly time, feeder movements and PCB table movements. The main difference between our CDPP model and the previous DPP (and EDPP) is that our CDPP calculated the robot arm movement distance as the maximum of the movement in Y or the movement in X (a chebychev distance) since our robot arm can move in X-axis and Y-axis concurrently, whilst the previous DPP (and EDPP) calculated the robot arm movement as a euclidean distance. This CDPP robot motion control has shown an improvement (in terms of assembly cycle time, *CT*) compared to Wang's DPP approach.

Therefore, we further explore the CDPP approach by introducing a triple objective function to improve the feeder setup in order to gain even better results. The function aims to minimise the *CT* together with minimising the feeder carrier and PCB table movements. By using this strategy we might be able to find better quality solutions with a better movement of feeder carrier and

PCB table. In summary, minimising the triple objective function strategy might provide better quality solutions (for feeder setup) compared to the strategies of minimising the *CT* and minimising the exchange frequency.

However, the CDPP approach is only applicable for the SMD placement machine that has a movable feeder carrier and PCB table with the robot arm is movable in *X*-axis and *Y*-axis concurrently.

The next chapters (i.e. chapter 5, 6, 7 and 8) investigate the optimisation of a theoretical multi-head SMD placement machine.

Chapter 5

An Investigation of an On-line Scheduling Approach for Multi-Head Surface Mount Device Placement Machine

5.1 Introduction

A lot of work has been done on improving the efficiency of SMD (surface mount device) placement machines; for example Ahmadi et al. (1988), Ahmadi and Mamer (1999), Altinkemer et al (2000), Deo et al. (2002), Ellis et al. (2002) and Tirpak et al. (2000). However, most previous work involves an off-line (predictive) scheduler. Usually, the off-line scheduler requires a long time to produce a good quality schedule. Unfortunately, most assembly lines operate in a dynamic environment. The predictive schedule is ineffective if there is a change in the resources after the schedule is generated. For example, if the component feeders are misallocated by the machine's operator or if some components are missing from the feeder carrier (e.g. they run out), then the solution given by an off-line scheduler becomes infeasible. Hence, the placement machine will be idle, waiting for a new schedule to be generated. The duration of the idle state will be dependent on the time taken by the off-line scheduler.

This chapter proposes a conceptual methodology for on-line scheduling to sequence the pickup-and-placement of component on a theoretical multi-head SMD placement machines in printed circuit board (PCB) assembly to overcome these spontaneous circumstances. Some of the latest technology in placement machines such as the MY19 (Mydata, 2002) and the HP-110 (Dima, 2003) have smart feeder carrier(s) that can automatically detect the exact availability and location of each component type on the feeder slot. For example, the HP-110 (a new DIMA machine), allows a feeder changeover while the machine is running and does not insist on a fixed feeder location (i.e. we can place/assign the component feeders at any feeder slot). The HP-110 uses a feeder barcode to detect the exact feeder location and an independent feeder counter (embedded within the tape feeder). We propose using this feature to enhance the scheduling of the PCB machine. Indeed, our on-line scheduling might eliminate the machine's idling time by starting the pickup-and-placement operations immediately after the PCB (and the PCB data) have been loaded into the machine and the machine might run continuously even if there are missing components or a feeder changeover occurs.

Assuming that the components on the feeder carrier may be misplaced or some of the required components are missing, we generate an initial schedule using a greedy constructive heuristic by only considering the available placement points. The initial solution can immediately be used to assemble components for the first PCB. While the placement machine is assembling components, we employ the CPU free time (whilst the robot arm is moving) to improve the initial schedule by using a simple descent search technique. Thus, the subsequent PCB's will use the improved schedule.

Based on a number of discussions with PCB assembly companies, they usually prefer not to change the feeder layout (feeder setup) too frequently, unless it is necessary. Therefore, an on-line scheduler is a solution. Moreover, the on-line scheduler allows them to place the components at any slot. Indeed, there is no component misallocation problem. The work presented in this chapter has been disseminated as follows:

Ayob, M. and Kendall, G. (2003a). Real-time scheduling for multi headed placement machine. *Proceedings of the 5th IEEE International Symposium on Assembly and Task Planning, ISATP'03*, Besançom, France, 9-11 July, 128-133.

5.2 Multi-Head Surface Mount Device Placement Machine

In this chapter and the subsequent two chapters (i.e. chapters 6 and 7), we study a component pick-and-place sequencing problem for a theoretical multi-head SMD placement machine. As described in chapter 3 (section 3.2.4), this type of machine has a fixed feeder carrier, a fixed PCB table and a positioning arm head that is equipped with a number of pipette/nozzles that are used to grasp the components. The feeder carrier consists of several feeder slots where the components are located. The PCB table holds the board in a locked position during a pick-place operation. The placement head is movable simultaneously in the X-Y direction.

A sub tour (we refer to a sub tour to differentiate from an overall tour, which is an operation to place all the required components onto a single board) means an operation taken by the robot arm to pick up and place a number of components (depending on the number of pipette/nozzles per head) in a single trip. A sub tour of the heads begins by picking up a number of components from the feeder. Then, it travels in an X and Y direction (simultaneously) and positions itself at the point where the component will be mounted. Then the pipette moves down (Z-direction) and mounts the component on the board before returning to its original position and repeating these steps for the next locations on the board that have to be mounted on the same sub tour. After completing a sub tour, the head returns to the feeder location to begin another sub tour.

There are many factors involved in determining the efficiency of pick-place operations of multi-head SMD placement machines such as the grouping of PCB points (also referred to as placement points) to a sub tour, pipette/nozzle assignment, pickup-and-placement sequencing etc. As the robot arm is equipped with a number of pipettes, the problem is to determine the sets of PCB points that will be visited by the robot arm (i.e. to place a component) in the same sub tour. For example if the robot arm is equipped with 8 pipettes, we may have 8 pickup points and 8 placement points in a sub tour. A sub tour of the robot arm begins by picking up a number of components (from the feeders) sequentially/concurrently and then travelling concurrently in an X-Y direction to place the components onto the PCB sequentially/concurrently. This procedure suggests a number of scheduling problems. For example:

a) Assigning the pickup pipette. The robot arm has a number of pipettes. In this work we assume all components are the same size (we ignore nozzle size selection i.e. no nozzle change operations and the nozzles are attached to the pipettes before the production starts). The issue is to determine which pipette/nozzle should be used to pickup a component such that we minimise the robot arm travelling distance. We must ensure that the PCB points will receive the correct component type. Therefore, if pipette/nozzle A picks up component type X and pipette/nozzle B picks up component type Y, then pipette/nozzle A must place component X at a placement point, which is expecting a component of type X in the sub tour (similarly with pipette/nozzle B). However, if both pipette/nozzle A and B pick up a component type X then the sub tour scheduling is easier as either pipette/nozzle A or B can be used to place X at a relevant point on the PCB. As the pipette/nozzles are located at a fixed position at the end of pickup/placement heads, the cost of picking up the next component is dependent on the pipette/nozzle used, the current location of the head and the current pipette/nozzle used to pick up the current component.

b) Sequencing the component pickups. The problem is to determine the sequence of picking up components in a sub tour to optimise the pickups.

c) Sequencing the placement operation. The problem is to determine the sequence of placing components in a sub tour to optimise the placements.

d) Assigning the placement pipette. Again, the issue is to optimise the placement and we must ensure that the PCB points will receive the correct component type. This problem is almost the same as problem (a). However, in this problem, the placement pipette/nozzle is assigned by aiming to minimise the placement operation (ignoring the cost of the pickup operation). Similarly, in problem (a) the assignment of pickup pipette/nozzles is done based on minimising the pickup operations (ignoring the cost of the placement operation).

e) Assigning PCB points to a sub tour.

f) Sequencing the sub tours. The aim is to optimise the sequence of sub tours in order to minimise the cycle time.

The aim of optimising the pickup-and-placement operations is to minimise the cycle time (CT). However, there is a trade-off between optimising the pickup and optimising the placement. Indeed, these sub problems are tightly intertwined. The problem requires various neighbourhood structures. Therefore, generally, local search approaches such as simulated annealing and tabu search have difficulty in solving this problem since they are dealing with single heuristics whilst the complexity of the problem requires many heuristics for exploring various neighborhood structures. Therefore, heuristics that are capable of exploring various neighborhood structures such as hyper-heuristics (Burke et al., 2003) and Variable Neighbourhood Search (Hansen and Mladenović, 2001) might be suitable for solving this type of problem. Hence, we study these two approaches in chapters 6 and 7, respectively.

5.3 On-line Scheduling Issues

Manufacturing operations, especially machine operations, are dynamic in nature and are subject to various interruptions, which may cause the predictive schedule to become ineffective or infeasible. Therefore, many scheduling papers in the manufacturing environment involve dynamic scheduling; for example Adzakpa et al. (2004), Cowling and Johansson (2002), Ouelhadj (2003), Sabuncuoglu and Bayiz, M. (2000), Sabuncuoglu and Karabuk (1999), Sun and Xue (2001) and Vieira et al. (2003), to name a few. Extensive surveys on dynamic scheduling can be found in Ouelhadj (2003) and Sabuncuoglu and Bayiz, (2000). Unfortunately, most research for the component pick-and-place sequencing problem in PCBA (printed circuit board assembly) does not focus on dynamic scheduling even though there are many spontaneous events such as running out of components, component feeders misallocated, defective components and dynamic feeder reloading during machine operation. This is due to the weakness of old technology SMD placement machines, that do not allow dynamic feeder reloading, unable to detect feeder misallocation etc.

On-line scheduling can be defined as reactive or dynamic scheduling where the schedule is not generated in advance, and decisions are made locally in real-time (Ouelhadj, 2003; Sabuncuoglu and Bayiz, 2000). In fact, on-line scheduling is different from predictive-reactive scheduling. Generally, the majority of the literature (Cowling and Johansson, 2002; Ovacik and Uzsoy, 1994; Sun and Xue, 2001; Vieira et al., 2003) on dynamic scheduling refers to predictive-reactive scheduling (rescheduling or reactive scheduling). In predictive-reactive scheduling, the schedule is generated in advance and will be modified or regenerated if disruption occurs during its execution (Ouelhadj, 2003; Sun and Xue, 2001; Sabuncuoglu and Karabuk, 1999). For example, Sun and Xue (2001) employed match-up and agent-based collaborative approaches to partially modify the originally created schedule for enhancing the reactive schedule efficiency when the schedule cannot be completed due to production changes such as a change of production orders or machine breakdown.

On-line dispatching rules (such as first-in-first-out, short processing time, priority rule and due date) are the most frequently used approaches in on-line scheduling (Ouelhadj, 2003; Sabuncuoglu and Bayiz, 2000). However, due to the myopic (short sighted) nature of the rules, the solution quality is sacrificed because they do not use global information. Kutanoglu and Sabuncuoglu (2001) argued that there is no single rule that yields the best performance over all conditions. Hence, changing a dispatching rule based on a current or future event can lead to a better performance than using a single rule (Kutanoglu and Sabuncuoglu, 2001). However, in the context of the component pick-and-place sequencing problems in PCB assembly, it is difficult to employ any dispatching rules. There is no pickup or placement priority unless we are dealing with small or large sized components that must be placed close to each other or there are multi-level components. In this case we may need to place a smaller component first. If we placed the larger component first, the nozzle may not be able to place the smaller due to the restricted space. Similarly, we may need to place a small component first so that a larger component can be placed over the top (i.e. multi-level placement). However, since most of the components on the PCB are small in size, this problem can easily be solved, by assigning larger size components with a lower priority than the small size components. Unlike the job shop scheduling problem in manufacturing industry where each job has a due date and needs to be processed on one or more machines and usually requires a reasonably long processing time, the component placement sequencing problem is concerned with of finding a good sequence for the robot arm (placement head) to pick up and place components onto the PCB. Indeed, each PCB type is usually produced in high volumes. Moreover, the scheduler for the placement machine is embedded into the machine that allows the scheduler to directly gather information about the current machine operation. Therefore, the scheduler can easily change the schedule whenever necessary.

Because of the technological characteristic of the SMD placement machine, the on-line scheduling approach is seem to be the most appropriate approach to schedule the component pick-and-place operations.

5.4 On-line Scheduling for Multi-Head Surface Mount Device Placement Machine

As the SMD placement machine is usually a microprocessor based system, the information about the missing and reloaded component types can be managed by interrupt service routines (ISR). An interrupt is a real-time event that can occur at anytime. The interrupt signal can automatically be generated by hardware or software. If the interrupt signal is triggered then the appropriate ISR will be activated in response to that event. The ISR is a special routine dedicated to each interrupt event. The ISR and the scheduler are separate/independent processes. The ISR will update the appropriate information to be used by the scheduler to regenerate/modify/improve the schedule.

In this theoretical framework, we suggest two interrupt events for this problem, the missing component's interrupt and the reloaded component's interrupt. The missing component's ISR (named as ISR1) is activated whenever there is a missing component type. Similarly, when there is a newly reloaded component, the reloaded component's ISR (named as ISR2) will be executed. ISR1 only needs to store the name of the missing component type in a missing component file such that the scheduler program can use this file to identify the missing component type. Similarly, ISR2 will store the name of a new reloaded component type into the component's reloaded file. ISR2 will also remove the name of component type that has just been reloaded from the missing component file. The scheduler will automatically remove the contents of the component's reloaded file after these component types have been inserted into the schedule. To prevent a 'system crash' or a race condition, the component's reloaded file and the missing component's file have to be defined as critical sections. As such, only one process can access them at a given time. The decision whether to proceed with the next PCB by moving out the uncompleted current PCB due to missing components' type or waits until all the missing components been reloaded and completing the current PCB, is dependent on the managerial policy. However, in practice, it is preferable to complete each PCB in the same day because the solder paste tends to dry up.

While the robot arm is moving to pickup a component(s), the machine's CPU might be in an idle state. After picking up the component(s), the robot arm will interrupt the CPU to acknowledge job completion. Then the CPU will send other control data to the robot arm. While the robot arm is moving to place a component (for example), the CPU may carry out other tasks, for example, component recognition and component alignment and then go into an idle state again. However, in general terms, we can say that the CPU is always in an idle state while the robot arm is moving or picking and placing components. Since the robot arm is normally an interrupt driven I/O device, we might make use of the CPU free time to do other work. When the robot arm completes its current task, it will send an interrupt request signal to the CPU. In responding to this interrupt request, the CPU will suspend the current running task (the task we will run during CPU free time). The appropriate interrupt service routine (ISR) will be called to service the request (which may include sending other appropriate control data). When the ISR completes the task, the CPU will automatically continue the suspended task. Information about interrupts and how they work can be found in many operating systems' textbooks such as (Silberschatz et al., 2000; Tanenbaum, 1992). In this work, we introduce an on-line scheduler that might utilise the CPU free time. We assume the components are randomly assigned to the feeder slots (or the operator may have misplaced the components). We generate a greedy constructive heuristic for a pick-and-place on multi-head placement machine that randomly assigns components to feeder slots, a placement point to a sub tour, a pipette/nozzle for pickup-and-placement, a pickup sequence, a placement sequence and sub tour sequences. In our proposed theoretical framework, we might start the pick-and-place operation immediately, once the initial schedule is ready (generated by the constructive heuristic). The first PCB will be processed using the initial schedule. While the robot arm is moving, picking and placing components, we may employ the CPU spare time to improve our initial schedule. We apply a local search, using a simple descent method, that only accepts an improved solution. We apply swaps between placement/pickup points in a sub tour or among the sub tours. The details of the swapping method are discussed in section 5.6.

While the CPU is running the optimisation software (our on-line scheduler) to find a better schedule, the robot arm may interrupt the CPU whenever it needs attention. Therefore, our optimisation software might run without incurring 'significant cost' to the assembly cycle time of the PCB being processed (we might pay a very small cost for ISR management). Thus, without paying 'any significant cost', we might improve the initial solution. By using this theoretical on-line scheduling concept, the pickup-and-placement schedule might always be updated and we may obtain a very good quality solution if the placement machine produces a large number of PCBs. Moreover, if the component feeder runs out of components, a new schedule may be easily generated without halting production. For the purpose of optimisation software, we do not have to concern ourselves with being too accurate about CPU spare time. As a rule of thumb, any heuristic might be applicable for the proposed on-line scheduling approach as long as it is not computationally expensive (capable of producing a result within few milliseconds) in order to avoid machine's delay (i.e. waiting for the schedule). The exact amount of time that can be allocated for the scheduler to generate, improve and/or repair the schedule is machine dependent (and problem dependent). In principle, any electronic equipment such as an SMD placement machine, that is a microprocessor-based system equipped with many mechanical parts, is usually in an idle state while the mechanical parts are performing their tasks. Indeed, sometime each mechanical part has an embedded controller. Again, this is a machine dependent issue. The latest technology placement machines also allow components to be reloaded during production (Mydata, 2002; Dima, 2003). After components have been reloaded, a new schedule must be generated and again our proposed theoretical on-line scheduling approach is capable of solving this problem.

5.5 The Scheduling Model

In this work, a pickup-and-placement time function for a theoretical multi-head SMD placement machine is developed to evaluate the proposed approach. We model a theoretical multi-head placement machine that has a single head equipped with *G* number of pipettes/nozzles, fixed PCB table and fixed feeder carrier. The machine has an arm and a head that can move concurrently in the X-Y axis. The objective function only needs to consider minimising the total assembly cycle time by minimising the travelling distance of the robot to perform pick-and-place operations since the PCB table and feeder carrier are not moveable. The following notation is used to describe the scheduling model (some notational differences are used from those in chapter 4 due to the different machine types):

CT	:	*the assembly cycle time to assemble all components;*
N	:	*the number of PCB points on the PCB;*
Q	:	*the total number of available PCB points to be scheduled, where* $Q \leq N$.
K	:	*the number of component types (each feeder slot holds multiple copies of one component type);*
G	:	*the number of pipette/nozzles per head;*
B	:	*the total number of sub tours;*
M	:	*the total number of feeder slots where* $K \leq M$;
$c(j,h)_{x,y}$	:	*the X,Y coordinate on the PCB, which will have a component placed there in the* h^{th} *placement sequence of the* j^{th} *sub tour;*
V	:	*the robot speed (average);*

λ	:	*the time for picking up a component;*
θ	:	*the time for placing a component;*
r	:	*the r*th *slot number where* $r \in \{0,1,2,\ldots,(M-1)\}$;
i	:	*the i*th *component type where* $i \in \{1,2,\ldots,K\}$;
k	:	*the k*th *pickup sequence in a sub tour where* $k \in \{1,2,\ldots,G\}$;
h	:	*the h*th *placement sequence in a sub tour where* $h \in \{1,2,\ldots,G\}$;
j	:	*the j*th *sub tour number where* $j \in \{1,2,\ldots,B\}$;
$I(j,h)$	:	*the time taken for the robot arm to travel from feeder carrier to PCB point and place a component in the h*th *placement sequence of the j*th *sub tour;*
$P(j,k)$	:	*the time taken for the robot arm to travel from PCB point to feeder carrier and pick a component in the k*th *pickup sequence of the j*th *sub tour;*
$\Phi_i(j,k)$	:	*the pipette/nozzle used to pick i*th *component in the k*th *pickup sequence of the j*th *sub tour;*
$\Omega_i(j,h)$	:	*the pipette/nozzle used to place i*th *component in the h*th *placement sequence of the j*th *sub tour;*
$S(r)$	:	*the r*th *slot distance referring to the origin of feeder slot,* $r=0$ *where* $S(0)=0$ *and* $r \geq K$;
$R(j,k)$	:	*the slot distance for the k*th *pickup sequence of the j*th *sub tour;*
$d(j,h)$	:	*the* $\max\{\lvert d(j,h)_x \rvert, \lvert d(j,h)_y \rvert\}$ *where x and y is the X,Y robot travelling distance to place a component in the h*th *placement sequence of the j*th *sub tour, where the distance is measured as a Chebychev distance (dictated by the maximum of X or Y travelling distance as the robot arm moves concurrently in X-Y axis);*
$m(j,k)$	:	*the* $\max\{\lvert m(j,k)_x \rvert, \lvert m(j,k)_y \rvert\}$ *where x and y is the X,Y robot travelling distance to pick a component in the k*th *pickup sequence of the j*th *sub tour, where the distance is measured as a Chebychev distance ;*
$z(j,k)$	:	*a decision variable where* $z(j,k)=1$ *if there is a pipette/nozzle assigned to pick or place a component in the k*th *sequence of j*th *sub tour, or '0' otherwise;*
$b_r(j,k)$	:	*a decision variable where* $b_r(j,k)=1$ *if there is a component to be picked up from feeder slot r in the k*th *pickup sequence of the j*th *sub*

		tour, or '0' otherwise;
F	:	*the gap between each two adjacent feeder slots;*
L	:	*the gap between each two successive (adjacent) pipette.*

The objective function is to minimise the assembly cycle time, CT:

$$\text{Minimise} \quad CT = \sum_{j=1}^{B}\left[\sum_{k=1}^{G} P(j,k) + \sum_{h=1}^{G} I(j,h)\right] \tag{5.1}$$

subject to:

$$\Phi_i(j,k) \in \{0,1,2,..(G\text{-}1)\};\ \ \Phi_i(j,k) \neq \Phi_e(j,l) \text{ if } k\neq l;\ \ j=1,2,..,B;\ \ k = 1,2,..,G; \tag{5.2}$$

$$\Omega_i(j,h) \in \{0,1,2,..(G\text{-}1)\};\ \Omega_i(j,h) \neq \Omega_e(j,l) \text{ if } h\neq l;\ \ j=1,2,..,B;\ \ h =1,2,..,G; \tag{5.3}$$

$\Phi_i(j,k) = \Omega_i(j,h)$ *Ensure a correct pipette/nozzle use to place a component type.* **(5.4)**

$$\sum_{k=1}^{G} z(j,k) \leq G, \forall j; \tag{5.5}$$

$\sum_{r=0}^{M-1} b_r(j,k) \leq 1, \forall j, \forall k;$ *Only one component can be picked up in each pickup sequence.* **(5.6)**

where:

$$B = \begin{cases} Q/G \ \text{ if } Q \ \ MOD \ \ G=0; \\ 1 + \ Q/G \ \text{ if } Q \ \ MOD \ \ G\neq 0; \end{cases} \tag{5.7}$$

$$P(j,k) = \left(\frac{m(j,k)}{V} + \lambda\right) * z(j,k); \tag{5.8}$$

$$I(j,h) = \left(\frac{d(j,h)}{V} + \theta\right) * z(j,k); \tag{5.9}$$

$$m(j,k)_x = \begin{cases} R(j,k) - \Phi_i(j,k)*L; & \text{if } j=k=1; \\ [R(j,k) - \Phi_i(j,k)*L] - [c(j\text{-}1,G)_x\text{-}\ \Omega_e(j\text{-}1,G)*L]; & \text{if } k=1, j>1; \\ [R(j,k) - R(j,k\text{-}1)] - [(\Phi_i(j,k)\text{-}\ \Phi_e(j,k\text{-}1))*L]; & \text{if } k>1; \end{cases} \tag{5.10}$$

$$m(j,k)_y = \begin{cases} 0 - [c(j\text{-}1,G)] & \text{if } k=1, j>1; \\ 0 & \text{otherwise} \end{cases} \tag{5.11}$$

$$d(j,h)_x = \begin{cases} [c(j,h)_x - R(j,G)] - [(\Omega_i(j,h) - \Phi_e(j,G))*L] & \text{if } h=1; \\ [c(j,h)_x - c(j,h-1)_x] - [(\Omega_i(j,h) - \Omega_e(j,h-1))*L] & \text{if } h>1; \end{cases} \quad \textbf{(5.12)}$$

$$d(j,h)_y = \begin{cases} c(j,h)_y\,; & \text{if } h=1; \\ [c(j,h)_y - c(j,h-1)_y]; & \text{if } h>1; \end{cases} \quad \textbf{(5.13)}$$

$$S(r) = r * F \quad \textbf{(5.14)}$$

$$R(j,k) = \sum_{r=0}^{M-1} \left[S(r) * b_r(j,k) \right] \quad \textbf{(5.15)}$$

The assembly cycle time, *CT* is a function of the robot travelling distance divided by the robot speed, plus a components' pickups (λ) and placements'(θ) time. We ignore other optimisation factors such as nozzle changeover, component feeder transportation time, simultaneous (gang) pickup, tray feeder reloading time etc. Since the robot (i.e. the arm and head of SMD placement machine) can move simultaneously in the *X-Y* axis, the robot travelling distance is dictated by the maximum of *X* or *Y* travelling distance, i.e. a Chebychev distance. In our formulation, we consider the robot makes a positive travelling distance when it moves in increasing *X* or *Y* coordinate and a negative travelling distance otherwise. As the origin of our coordinate system is referring to a feeder slot *r=0,* then the robot makes a negative travelling distance to move from a PCB point for picking up the first component of a sub tour from feeder carrier (equation 5.11). A complete tour of a robot consists of *B* sub tours and each sub tour has at most *G* pairs of pick-and-place points (equation 5.1). The constraint 5.2, 5.3 and 5.4 ensures that each PCB point will receive a correct component type. Whereas, constraint 5.5 ensures that at most, *G* components can be picked up in a sub tour. At any given time, only one component can be picked up from a component feeder (constraint 5.6). If there exist some components of the same type in a sub tour, then the robot has more flexibility to pick-and-place the component type.

5.6 Implementation

Initially, the algorithm will read all the PCB data and identify the PCB points that it can immediately schedule (there could be PCB points that cannot be scheduled at the moment because of components being missing from the feeder carrier; these will be marked and inserted into a later schedule, whenever their components are reloaded onto the feeder carrier).

Our random constructive heuristic begins by randomly assigning each PCB point to a sub tour. The size of a sub tour depends on the number of available pipettes/nozzles per head and the number of available placement points. Each sub tour consists of a set of pickup points and a set of placement points. The pipettes/nozzles are indexed from 0 (left most side) to *G-1* (right most side). For each sub tour, the pickup-and-placement pipettes/nozzles are randomly allocated. However, we must ensure that each placement point is assigned the correct component type. The pickup-and-placement sequences in each sub tour are also generated randomly and are independent of each other. A sub tour begins by picking up a set of components in a random sequence, then placing these components onto the PCB, also in a random placement sequence (but the correct PCB point will receive a component of the correct type). Finally, the constructive heuristic randomly sequences the sub tours to complete the overall pickup-and-placement schedule to generate an initial solution.

To improve the initial solution, we apply a local search with a simple descent method. In the local search we only accept improved solutions. Once an improved neighbouring solution is found, we then accept the new solution and continually search around the neighbourhood of the new solution until the stopping criterion is met. We propose six swapping methods, which are:

1) L1: Swap the pickup sequence in a sub tour. For each sub tour we perform *G* number of swaps in the component pickup sequence. In a sub tour, each i^{th} component pickup sequence will be swapped with a randomly selected j^{th} component pickup sequence.

2) L2: Swap the placement sequence in a sub tour. For each sub tour we perform *G* number of swaps in the component placement sequence. In a sub

tour, each i^{th} component placement sequence will be swapped with a randomly selected j^{th} component placement sequence.

3) L3: Swap the pickup pipettes/nozzles in a sub tour. For each sub tour we perform G number of pipette/nozzle swaps of pickup operations. In a sub tour, the pipette's/nozzle's used in the i^{th} component pickup sequence will be swapped with a randomly selected pipette/nozzle used in the j^{th} component pickup sequence. If the swapping operations involve two different component types, then we modify the appropriate placement pipette/nozzle in the sub tour such that the PCB points will receive the correct component type.
4) L4: Swap the placement pipettes/nozzles in a sub tour. For each sub tour we perform G number of pipette/nozzle swaps in placement operations. In a sub tour, the pipette's/nozzle's used in the i^{th} component placement sequence will be swapped with a randomly selected pipette/nozzle used in the j^{th} component placement sequence. If the swapping operations involve two different component types, then we modify the appropriate pickup pipette/nozzle in the sub tour such that the component will be picked up with the right pipette/nozzle. The neighbours in this neighbourhood are obtained in a similar way to that, which we use to obtain the pickup pipette's/nozzle's neighbourhood except that instead of swapping pickup's pipette/nozzle, we swap the placement's pipette/nozzle. The different effect occurs when a PCB point can be placed by several pipettes/nozzles (i.e. in a sub tour, there are several pipettes/nozzles holding the same component type). In this case, the pickup cost cannot be affected by swapping the placement pipette/nozzle.
5) L5: Swap the PCB placement points among sub tours. In this swapping method, we perform Q number of swapping operations where Q is a total number of available placement points. Each placement point in a sub tour will be swapped with other placement points in another sub tour (chosen at random). If the swapping operations involve two different component types, then we modify the appropriate pickup component in the appropriate sub tour such that the pickup components are valid in both sub tours.

6) L6: Swap the sub tours sequence order. We perform *S* number of swapping operations where *S* is a total number of complete sub tours.

There are many factors involved in determining the quality of the solution such as the grouping of placement points to a sub tour, pipette/nozzle assignment, pickup-and-placement sequencing etc. Usually, optimising one factor will increase the cost of another factor(s). For example, if we optimise the grouping of the placement points to a sub tour, we may have to pay a cost in pickup time. Since there is no clue or good strategy on selecting the best swapping method at any point, we use a random selection approach that is, the next local search swap is chosen by generating a random number between 1 and 6 (since there are six swapping methods). This process continues until completing the number of iterations, which is set at 100 for our experiments.

5.7 Testing and Results

In our experiments we assume that the feeder carrier and PCB table are positioned close to each other in order to minimise the robot arm travel distance (Su et al., 1995). We also assume that the gap between the feeder carrier and the PCB board is 10 unit length, the pipette's/nozzle's gap is equal to the size of the feeder slots (chosen as 4 unit length), all components are the same size (we ignore the nozzle size selection) and there are no defective components. We assume that all components use the same nozzle type and the speed of robot arm is constant for all component types. In the experiment we modelled the theoretical multi-head placement machine as a head equipped with 4 pipettes/nozzles. We further assume that the SMD placement machine can only pickup one component at a time but the number of components that can be picked up in a sub tour is dependent on the number of pipettes/nozzles per head (no simultaneous pickup). Since we modelled the placement machine that has a fixed PCB table and a fixed feeder carrier, we only apply five of the seven factors (table 5.1) of the parameters used by Su et al. (1995). The other two factors, feeder carrier speed and PCB table speed, are not required since we model the fixed feeder carrier and fixed PCB table SMD placement machine.

TABLE 5.1: EXPERIMENTAL PARAMETERS

Factors	**Levels (low/high)**
Number of assembly points (N)	20/100
Number of component types (K)	8/20
Length of PCB (BL)	100 (unit distance)
Width of PCB (BW)	50 (unit distance)
Speed of robot (V)	10 (unit distance/unit time)

The pick up, λ and placement time, θ are both set as 0.5 unit time. For the purpose of generating the random placement points, we set the length and the width of the PCB as 100 and 50 unit length, respectively, such that the random placement points fall within these limits. The placement points are generated randomly. Since we use a constructive heuristic that randomly groups the PCB points into sub groups (each sub group will be assigned to a sub tour), randomly assign component feeders to slots, randomly sequence the component pickup-and-placement, randomly assign pipettes/nozzles for pickup-and-placement and randomly sequence the sub tours, we can use the same dataset for different runs (every run will generate different initial solutions with different feeder setup, which can be considered as different problem instance since we only focusing on sequencing the component pick-and-place operation). Hence, two datasets that are randomly created are adequate to demonstrate our approach. Dataset 1 has 20 assembly points consisting of 8 component types whilst dataset 2 has 100 assembly points consisting of 20 component types. We chose these two datasets in order to show that our method can work with a larger problem size as well as a smaller problem. To simulate the assembly cycle time, we set the speed of the robot arm, V as 10 unit distance/unit time (as shown in table 5.1).

We ran the experiment using an AMD Athlon XP1700+ PC with 1.47GHz speed and 240 MB RAM. The computational results are shown in table 5.2 and table 5.3 and are obtained from 10 runs for each dataset.

The results in table 5.2 and table 5.3 show that our greedy constructive heuristic can generate an initial solution in a short time of about 0.070 seconds for dataset 1 and 0.236 seconds for dataset 2. For dataset 1, the initial solutions have improved about 36.42% after 10.22 seconds (average result) whilst a more complex dataset gained 43.19% after 75.703 seconds (average result). The results show that a good initial solution does not guarantee a good final solution and a bad initial solution does not mean that we cannot obtain a good final solution. For example, in dataset 1, test 2 started with a CT=54.23 unit time as an initial solution and finished with a CT=28.20 unit time as a final solution whilst test 5 started with CT=49.11 unit time but finished with CT=38.54 unit time. In dataset 2, test 1 started with a CT=419.15 unit time as an initial solution and finished with a CT=212.29 unit time as a final solution whilst test 9 started with CT=367.15 unit time but finished with CT=236.54 unit time. Generally, we can see that a quality of a final solution is not dependent on the quality of an initial solution.

TABLE 5.2: AN EXPERIMENTAL RESULT OF TEN RUNS ON DATASET 1 (N=20, K=8)

Test	**Constructive Heuristic**		**Improvement Heuristic**		**Improvement (%)**
	CT	**P**	**CT**	**P**	
1	52.89	0.078	31.89	8.844	39.71
2	**54.23**	**0.062**	**28.20**	**10.093**	**48.00**
3	55.61	0.063	35.16	9.281	36.77
4	48.06	0.063	28.56	12.047	40.57
5	**49.11**	**0.078**	**38.54**	**9.031**	**21.52**
6	46.79	0.078	31.10	12.312	33.53
7	44.48	0.078	29.23	11.015	34.29
8	52.65	0.063	32.98	7.953	37.36
9	54.78	0.062	34.48	10.454	37.06
10	49.21	0.078	31.81	11.203	35.36
Average		**0.070**		**10.220**	**36.42**
			Standard deviation:		**6.66**

Note: *CT = Assembly cycle time (unit time)*
P = Computation time (seconds)

The results in tables 5.2 and 5.3 apparently show a higher variation in improvement for the smaller dataset (N=20, K=8) compared to the larger dataset

(N=100, K=20). As a rule of thumb, a smaller dataset has a smaller solution space compared to a larger dataset. As a result, a neighbourhood search in the smaller dataset quickly converges when compared to a search in the larger dataset. In this experiment, we have set 100 iterations as a termination criterion. The search in a smaller dataset may have already converged by 100 iterations, whilst at this iteration, there are still many unexplored neighbours in the larger dataset, which means that the chances of improving the solution quality at this iteration is still high for the larger dataset. Therefore, at 100 iterations, the search in the smaller dataset might already been trapped in local optima. Since we have used different problem instances for each run, the solution quality of the local optima might vary (even one problem instance has many local optima). This explains why we have obtained a small variation in improvement for the larger dataset.

TABLE 5.3: AN EXPERIMENTAL RESULT OF TEN RUNS ON DATASET 2 (N=100, K=20)

Test	Constructive Heuristic		Improvement Heuristic		Improvement (%)
	CT	P	CT	P	
1	**419.15**	**0.250**	**212.29**	**75.187**	**49.35**
2	387.52	0.234	209.15	80.266	46.03
3	391.46	0.234	222.40	76.797	43.19
4	413.59	0.219	236.86	79.109	42.73
5	397.95	0.219	217.60	76.312	45.32
6	394.26	0.235	221.05	87.422	43.93
7	426.06	0.250	246.10	71.375	42.24
8	363.50	0.235	218.86	59.359	39.79
9	**367.85**	**0.250**	**236.54**	**70.297**	**35.70**
10	379.00	0.234	213.63	80.907	43.63
Average		**0.236**		**75.703**	**43.19**
			Standard deviation:		**3.65**

5.8 Summary

Due to the technological characteristics of the SMD placement machine, the on-line scheduling approach seems to be the most appropriate approach to schedule the component pick-and-place operations. As the CPU of the SMD placement machine is usually in an idle state while the robot arm is moving, picking and placing components and the robot arm is normally an interrupt driven I/O device, we may make use of the CPU free time to improve the initial schedule. Hence, this chapter has introduced a theoretical on-line scheduling framework that could employ CPU free time to improve the initial schedule. We use a greedy constructive heuristic to generate an initial solution then apply a simple descent method to improve the initial schedule. Results based on the experimentation on the theoretical multi-head SMD placement machine showed that the CT improved by 36.42% (dataset 1) and 43.19% (dataset 2) over the initial schedule. By using this on-line scheduling concept, the pickup-and-placement schedule is continually updated and we obtain a very good quality solution if the placement machine produces a large number of PCBs. Moreover, if the component feeder runs out of components, a new schedule is easily generated without halting production. Results also indicate that generally the quality of the final schedule is not dependent on the initial schedule.

The work carried out in this chapter is an algorithmic approach on an abstract machine, which was tested on artificial datasets. The proposed theoretical on-line scheduling framework utilised the CPU free time to continuously improve the pick-and-place schedule of the next PCB, while the machine is performing the pick-and-place operation of the current PCB. For the purpose of optimisation software, we do not have to concern ourselves with being too accurate about CPU spare time. As a rule of thumb, any heuristic might be applicable for the proposed on-line scheduling approach as long as it is not computationally expensive (capable of producing a result within few milliseconds) in order to avoid machine's delay (i.e. waiting for the schedule). The exact amount of time that can be allocated for the scheduler to generate, improve and/or repair the schedule is machine dependent (and problem dependent). In principle, any electronic equipment such as an SMD placement machine, that is a microprocessor-based system equipped with many mechanical parts, is usually in an idle state while the mechanical parts are performing their tasks. Indeed, sometime each mechanical part has an embedded controller. Again, this is a machine dependent issue. By adopting the proposed approach into the real production line, one can benefit by continuously improving the machine throughput. Of course, some modifications may be needed to suit any specific machine constraints.

The next chapter investigates how a hyper-heuristic approach can be applied to further improve the efficiency of a theoretical multi-head SMD placement machine. In chapter 7, we then employ a variable neighbourhood search. The hyper-heuristic and the variable neighbourhood search approaches are both suitable to be integrated with an on-line scheduling approach.

Chapter 6

Hyper-heuristic Approaches for Multi-Head Surface Mount Device Placement Machine

6.1 Introduction

In the previous chapter we presented a theoretical framework for an on-line scheduler that could continually improve the component pick-and-place schedule by employing CPU free time to continually search for an improved schedule. This chapter studies a hyper-heuristic approach to further improve a theoretical multi-head surface mount device (SMD) placement machine. This approach can also be integrated with the proposed on-line scheduling in searching for a better quality schedule. A motivation for investigating a hyper-heuristic approach, instead of the other meta-heuristics, is that the hyper-heuristic framework provides a way to combine many heuristics when searching for good quality solutions. This feature is useful for solving the component pick-and-place sequencing problem of a multi-head SMD placement machine since this problem appears to benefit from having access to a number of heuristics. Most previous works on hyper-heuristics (such as Burke et al., 2003a, b; Cowling et al., 2001a, b, 2002a, b, c; Ross et al., 2002; Schulenburg et al., 2002) focus on sequencing the calls of the low-level heuristics (LLHs). They report successful results. However, in this work we investigate on improving the acceptance criteria. In general, the low-level heuristics (LLH) are simple local searches, k-opt operators or other heuristics that are problem-dependent. Of course, the sequence of LLH is important but in our problem domain we are dealing with the LLHs that randomly create the neighbour solutions. Since the neighbour's solution is randomly generated by the LLHs, we cannot measure the performance of each LLH based on the historical performance as the quality of the obtained solution does not represent the efficiency of the LLH. This statement is supported by the experimental results from this work where the Choice-Function described in (Cowling et al., 2001a) does not perform well in all test problems.

There are many determining factors in minimising the assembly cycle time of multi-head SMD placement machine such as the optimisation of the pickup sequence, the placement sequence, pipette/nozzle assignment, sub-tour grouping and sequencing the sub-tour. However, optimising one factor may increase the cost of another factor(s). The complexity of this problem causes a difficulty in devising a good strategy to minimise the assembly cycle time. The advantage of using a hyper-heuristic is their ability to combine a number of heuristics. By applying a hyper-heuristic approach, we do not have to concern ourselves with the trade-off between the optimisation of the important factors as this will be catered for within the hyper-heuristic. The work presented in this chapter has been disseminated as follows:

a) Ayob M. and Kendall G. (2003b). An investigation of an adaptive scheduling for multi headed placement machines. *Proceedings of the 1st Multidisciplinary International Conference on Scheduling: Theory and Applications, MISTA 2003*, Nottingham, UK, 13-16 Aug, 363-380.

b) Ayob, M. and Kendall, G. (2003c). A monte carlo hyper-heuristic to optimise component placement sequencing for multi head placement machine. *Proceedings of the International Conference on Intelligent Technologies, InTech'03*, Chiang Mai, Thailand, 17-19 Dec, 132-141.

6.2 Hyper-heuristic Background

Many heuristics are problem-dependent (Reeves and Beasley, 1995). Meta-heuristics are capable of producing good quality solutions, but they often involve an adjustment of the relevant parameters in order to be applied to a new problem or even different problem instances (Burke et al., 2003b; Aickelin and Dowsland, 2000). However, there are some heuristics that are not problem-specific such as hyper-heuristic (Burke et al., 2003a, 2003b; Cowling et al., 2001a, 2001b, 2002a, 2002b, 2002c) approaches that aim to be a general-purpose heuristic that can handle a wide range of problems. Hyper-heuristics are (meta-)heuristics that can operate on (meta-)heuristics (Burke et al., 2003a, 2003b). In other words, a hyper-heuristic is a heuristic, which chooses a heuristic among heuristics (Burke et al., 2003a, 2003b). The term 'hyper-heuristic' was introduced by Ross et al. (2000). However, the idea of hyper-heuristics, although not using the term, was first published by Fisher and Thompson (1963). They solved a job-shop scheduling by adopting a probabilistic weighting on the low-level heuristics as a learning mechanism.

The hyper-heuristic framework manages a set of low-level heuristics, which operates at a higher level of abstraction without having access to the problem domain-knowledge (Cowling et al., 2002b). Generally, most hyper-heuristic researchers claim that the hyper-heuristic is a generic, robust and easy-to-implement approach (Burke et al., 2003a, 2003b, 2005a, 2005b; Cowling et al., 2001a, 2001b, 2002a, 2002b, 2002c; Han and Kendall, 2003).

Previous work has attempted to combine simple heuristics (in a hyper-heuristic framework) using a choice-function (Cowling et al., 2002b), hyper-genetic algorithm (Hyper-GA) (Cowling et al., 2002a; Hart et al. 1998; Ross et al., 2003), tabu assisted hyper-heuristic genetic algorithm (hyper-TGA) (Han and Kendall, 2003), tabu-search hyper-heuristic (Burke et al., 2003b; Kendall and Mohd Hussin, 2003, 2004), case based (Burke et al., 2005b; Petrovic and Qu, 2002), reinforcement learning (Nareyek, 2003), great deluge (Kendall and Mohamad, 2004b) and simulated annealing (Bai and Kendall, 2003; Dowsland et al, 2005a, 2005b).

Cowling et al. (2001a, 2001b, 2002b) developed a choice-function hyper-heuristic that adaptively ranks the LLHs based upon the historical performance of individual LLHs in order to suggest the next LLH to apply. A choice-function consists of three elements; these are the recent effectiveness of the LLH, recent effectiveness of consecutive pairs of LLH and the amount of time since the LLH was last called. They used the approach to schedule a sales summit (Cowling et al., 2001a, 2001b), project presentations (Cowling et al., 2002b) and nurse rostering (Cowling et al., 2002c). Results show that the choice-function hyper-heuristic is capable of solving these three problems, thus demonstrating its ability to generalise across problem types.

The Hyper-GA hyper-heuristic (Cowling et al., 2002a) evolves the sequence of calling the LLHs by representing a gene in a chromosome with a LLH. The chromosome is evaluated by the quality of the solution obtained when applying the LLHs in the sequence denoted by the chromosome. Subsequently, Han and Kendall (Han and Kendall, 2003) extended the work in (Cowling et al., 2002a) by proposing a tabu assisted hyper-heuristic genetic algorithm (hyper-TGA) to solve a personnel scheduling problem. They employed a tabu list to forbid genes (LLHs), which did not perform well, from being called during the following *n* generations. Results showed that the new approach is superior to their previous approaches.

Burke et al. (Burke et al., 2003b) utilised a tabu-search hyper-heuristic to solve timetabling and nurse rostering problem. They applied rules based on the principle of reinforcement learning to select the appropriate heuristic to be applied at a given time. The LLHs are adaptively ranked based on their historical performance. The tabu-search hyper-heuristic maintained a tabu list (on a ‘First In-First Out’ basis) of heuristics to avoid choosing the heuristics from the tabu list. The idea is to prevent selection of underperforming heuristics that have been recently called. Heuristics that are tabu are released once the current solution is modified. They claimed that the tabu-search hyper-heuristic is capable of producing solutions that are competitive with other well established meta-heuristics.

Other works in the hyper-heuristic area can be found in (Bai and Kendall, 2003; Burke et al., 2002, 2003c, 2005a, 2005b; Dowsland et al., 2005a, 2005b; Gaw et al., 2004; Hart and Ross, 1998; Hart et al., 1998; Kendall and Mohd Hussin, 2003; Kendall and Mohamad, 2004b, c; Narayek, 2003; Petrovic and Qu, 2002; Ross et al., 2002, 2003; Schulenburg et al., 2002; Soubeiga, 2003). A general framework of hyper-heuristic is shown in figure 6.1.

The communication between the hyper-heuristic and the LLHs uses a standard interface. Only non-domain specific data such as the solution's quality and the computation time is allowed to cross the barrier between the hyper-heuristic and the LLHs. The hyper-heuristic only knows that it has a certain number of heuristics on which to operate and whether the objective function is being minimised or maximised. The general structure of the hyper-heuristic algorithm is shown in figure 6.2.

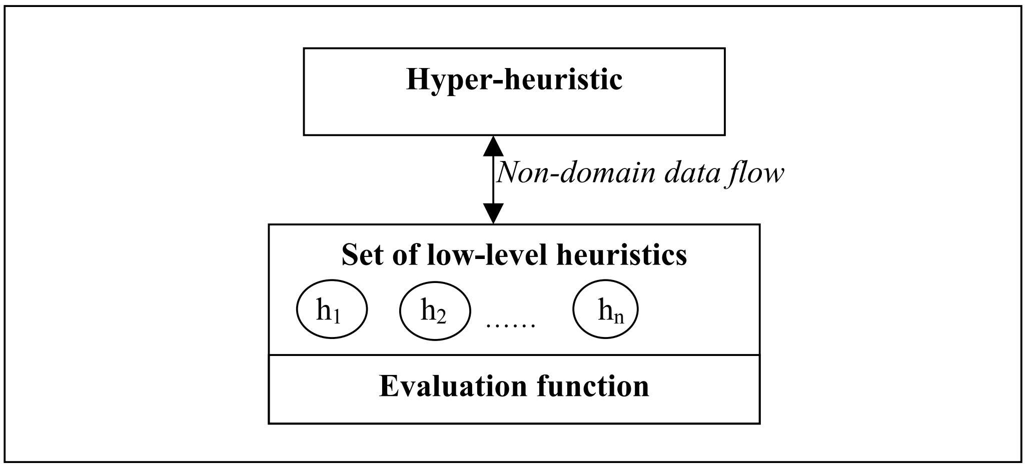


Figure 6.1: Hyper-heuristic framework

```
Step 1: (Initialisation)
    (A) Choose a starting solution $S_0 \in S$;
    (B) Define $H$ as a set of LLH;
    (C) Record the best obtained solution, $S_{best} = S_0$ and $F(S_{best}) = F(S_0)$;

Step 2: (Choice and termination)
    (A) Choose an $H_c \in H$;
    (B) Apply $H_c$ to produce $S_c \in N(S_0)$;
    (C) Compute $\delta = F(S_c) - F(S_0)$;
    (D) If the acceptance criteria is true, then accept $S_c$ (and proceed to
        Step 3);
    (E) If $S_c$ is rejected and stopping condition=false, then return to
        Step2(A);
    (F) Terminate by a stopping condition.

Step 3: (Update)
    Re-set $S_0 = S_c$, and if $F(S_c) < F(S_{best})$, perform Step1(C). Return to
    Step2 if stopping condition=false.
```

Figure 6.2: A general structure of a hyper-heuristic algorithm

6.3 Greedy Search Hyper-heuristic

6.3.1 Implementation

We develop two constructive heuristics that are a randomised and an ordered constructive heuristic. The difference between these two constructive heuristics is that the earlier one will randomly assign PCB points to a sub tour (as used in chapter 5) whilst the latter will sort the PCB points starting with the minimum of maximum (X,Y) coordinate (then with the minimum (X,Y) when duplication of maximum (X,Y) exists) and assigns the sorted PCB coordinate consecutively, starting at the top in the list, to a sub tour.

To create a neighbour schedule of a current schedule, we applied six swapping methods (called low-level heuristics) as proposed in chapter 5. Instead of using a simple random descent method in each local search (i.e. low-level heuristic), as used in chapter 5, we apply three neighbourhood search techniques to determine a move from one schedule to another schedule. These being:

a) A random descent (as used in chapter 5), which randomly selects a neighbouring schedule and accepts the first improved schedule. Next, the search process continues to a new neighbourhood space until it meets a stopping condition (e.g. completed a certain number of iterations). Using this method, the initial schedule may change a few times (in a sub tour every time the low-level heuristic is called) if we always find a better schedule during the searching process.

b) A random move that temporarily accepts a randomly chosen neighbouring schedule without considering the schedule's quality but always maintains the best schedule. In this approach, we temporarily move from one neighbourhood space to another space in every iteration but the schedule will only be updated if a better schedule is found.

c) A steepest descent that searches some neighbourhood and selects the best neighbour schedule. In this approach, we may make at most one move in a sub tour every time the low-level heuristic is called.

As in chapter 5 section 5.6, since there is no clue, or good strategy, on selecting the best swapping method at any point, we apply a greedy search method that randomly selects a low-level heuristic (L1 to L6) in each iteration by generating a random number between 1 (will select L1) and 6 (will select L6). The L1, L2, L3, L4, L5 and L6 were described in chapter 5 section 5.6. This process continues until completing a certain number of iterations (this is set to 100 in our experiments as in chapter 5). By introducing a greedy search approach at the higher level, we can diversify our search space and thus reduce the chances of getting trapped in local optima. Each low-level heuristic only searches around the specific neighbouring schedules and may get trapped in local optima. In fact, each low-level heuristic creates a different solution space that, possibly, cannot be achieved by another low-level heuristic. By switching from one low-level heuristic to another, using a greedy search, we may avoid getting trapped in local optima.

In order to analyse the effect of the two types of constructive heuristics, the three acceptance criteria approaches and the calling sequences of the low-level heuristics, we designed ten heuristics (obtained from a combination of the constructive heuristics, the acceptance criteria and the calling sequence method where in each combination, six low-level heuristics will be randomly or cyclically called and the move in each low-level heuristic will follow the acceptance criteria) as shown in table 6.1. Based on our preliminary experiment, we observed that the efficiency of the random descent and the steepest descent methods are almost the same in solving the component pick-and-place sequencing problem on multi-head SMD placement machine. We also found that a random move acceptance criterion cannot perform well. Therefore, we only developed ten heuristics instead of eighteen possible combinations, which ignored the combinations of random descent and random move acceptance criteria with deterministic 1 and deterministic 2 calling sequences. Actually, the idea of introducing the deterministic 1 and deterministic 2 calling sequences, are just to investigate the effect of calling sequence.

Deterministic 1 and 2 call the low-level heuristics using a predetermined sequence. Deterministic 1 cyclically calls L4, L3, L2, L1, L6 and L5 until it meets the stopping criteria whilst deterministic 2 cyclically calls L2, L4, L1, L3, L6 and L5 until it meets the stopping criteria. These sequences are chosen as it may be a good strategy to optimise within a sub tour before trying to optimise sub tours. Deterministic 1 will begin optimising nozzle assignment in a sub tour (L4 and L3), next optimising placement and pickup sequences in a sub tour (L2 and L1), and finally optimising sub tours (L6 and L5). Deterministic 2 will begin optimising placement sequences in a sub tour (L2), then optimising placement nozzle assignment in a sub tour (L4), next optimising pickup sequence in a sub tour (L1), then optimising pickup nozzle assignment in a sub tour (L3), and finally optimising sub tours (L6 and L5). By having these two deterministic heuristics and random greedy heuristic, we can observe the effect of sequencing low-level heuristic calls and randomising the calling sequence.

TABLE 6.1: HEURISTICS FOR MULTI-HEAD SMD PLACEMENT MACHINE

Heuristic ID	**Constructive Heuristic**	**Acceptance Criteria**	**Calling Sequence**
H1	random	random descent	random
H2	ordered	random descent	random
H3	random	random move	random
H4	ordered	random move	random
H5	random	steepest descent	random
H6	ordered	steepest descent	random
H7	ordered	steepest descent	deterministic 1
H8	ordered	steepest descent	deterministic 2
H9	random	steepest descent	deterministic 1
H10	random	steepest descent	deterministic 2

6.3.2 Testing and Results

We use the same assumptions as in chapter 5. In the experiment we modelled the same theoretical multi-head placement machine that has a head equipped with 4 or 8 nozzles that can pick no more than 4 or 8 components (dependent on the number of pipette/nozzles per head) in a sub tour. As in chapter 5, we only apply five experimental factors as shown in table 6.2.

TABLE 6.2: EXPERIMENTAL PARAMETERS

Factors	**Levels (low/high)**
Number of assembly points (N)	50/100/150/200
Number of component types (K)	10/20/30/40
Length of PCB (BL)	100 (unit distance)
Width of PCB (BW)	50 (unit distance)
Speed of robot (V)	10 (unit distance/unit time)

As in chapter 5, we set the length and the width of the PCB as 100 and 50 unit lengths, respectively. The placement points are generated randomly. Five datasets are randomly created to demonstrate our approach:

a) Dataset 1 has 100 assembly points and 20 component types;

b) Dataset 2 has 50 assembly points and 10 component types;

c) Dataset 3 has 100 assembly points and 20 component types;

d) Dataset 4 has 150 assembly points and 30 component types and

e) Dataset 5 has 200 assembly points and 40 component types, respectively.

We ran the experiments using an Intel® Pentium®4 PC with 1.5GHz speed and 256 MB RAM.

Two types of tests are conducted. Type 1 uses dataset 1 that has five runs starting with the same initial schedule and the same feeder setup. This experiment simulates a four nozzle (per head) SMD placement machine. We designed this test to show the consistency of the results and to make a fair comparison among the heuristics. The initial schedule was generated using an ordered constructive heuristic. We cannot perform this test on H1, H3, H5, H9 and H10 since they use the random constructive heuristic, hence will start with different initial solutions. The computational results of this test are shown in table 6.3.

The type 2 test uses the datasets 2, 3, 4 and 5, which varies the number of assembly points and component types. This experiment simulates an eight nozzle (per head) SMD placement machine. All the tests in each dataset will use the same feeder setup and the initial schedule is the same for heuristics H2, H4, H6, H7, H8 but differs in heuristics H1, H3, H5, H9 and H10 (as these heuristics use a random constructive heuristic). The test is designed to show how these heuristics act with a more complex problem and to show the consistency of the results obtained from type 1 test. Table 6.4 shows the results of this test.

TABLE 6.3: AN EXPERIMENTAL RESULT OF 5 RUNS ON DATASET 1

Heuristic ID	**Final assembly cycle time, CT (unit time)**					**CT_{avg}**	**I (%)**	**Dev**
	Test 1	**Test 2**	**Test 3**	**Test 4**	**Test 5**			
H2	194.6	197.2	205.6	206.5	203.1	201.4	40.7	5.24
H4	260.4	280.6	279.3	272.8	290.0	276.6	18.5	10.94
H6	209.7	216.4	201.9	202.6	206.0	207.3	38.9	5.96
H7	202.8	196.8	197.7	201.9	207.8	201.4	40.7	4.43
H8	201.3	214.5	201.0	197.9	197.8	202.5	40.3	6.91

Note: H2, H4, H6, H7, H8 = Heuristic ID; *CT_o=Initial CT =339.4 unit time;*
CT_{avg} = Average CT; *Dev=Standard deviation of CT*
*I = CT's Improvement=(CT_o-CT_{avg})*100/ CT_o*

The results in table 6.3 show that the heuristics H2, H6, H7 and H8 produce solutions, which are fairly equal in quality; these being 40.7%, 38.9%, 40.7% and 40.3% (respectively) improvement over the initial CT. Theses heuristics (H2, H6, H7 and H8) also have fairly equal standard deviation values; these being 5.24, 5.96, 4.43 and 6.91. This indicates that the efficiency of the random descent and the steepest descent methods are almost the same in solving the component pick-and-place sequencing problem on multi-head SMD placement machine. Results also show that the random greedy search heuristic works as well as the two deterministic heuristics. We can say that in this problem, the sequence order of calling low-level heuristics does not influence the quality of the final schedule. Perhaps, we can also state that most of our low-level heuristics have equal performance. On the contrary, heuristic H4, which accepts all moves but always keeps the best schedule as a final schedule, cannot perform as well as heuristics H2, H6, H7 and H8. H4 can only improve 18.5% over initial CT with higher variation in CT improvement (i.e. Dev=10.94). This shows that accepting all moves to find an improved schedule is not a good strategy in solving this problem. An average processing time to obtain a final schedule for the test is 34.06 seconds (for 100 iteration).

TABLE 6.4: AN EXPERIMENTAL RESULT OF TEN RUNS ON DATASETS 2, 3, 4 AND 5

	Dataset 2 (N50, K=10)			**Dataset 3 (N100, K=20)**			**Dataset 4 (N150, K=30)**			**Dataset 5 (N200, K=40)**			**Average I (%)**
	CT_o	**CT_{avg}**	**I(%)**	**CT_o**	**CT_{avg}**	**I(%)**	**CT_o**	**CT_{avg}**	**I(%)**	**CT_o**	**CT_{avg}**	**I(%)**	
H1	164.3	99.4	39.5	421.3	194.3	53.9	711.9	339.0	52.4	1031.7	499.1	51.6	49.3
H2	143.7	79.8	44.4	368.0	174.8	52.5	586.3	311.4	46.9	981.7	494.4	49.6	48.4
H3	170.5	136.0	20.3	424.6	333.2	21.5	686.7	548.8	20.1	1097.9	845.9	23.0	21.2
H4	143.7	116.4	19.0	368.0	263.3	28.4	586.3	501.5	14.5	981.7	833.4	15.1	19.3
H5	171.8	85.6	50.2	409.4	186.2	54.5	721.9	334.9	53.6	1031.0	508.2	50.7	52.3
H6	143.7	85.0	40.9	368.0	173.4	52.9	586.3	319.4	45.5	981.7	488.2	50.3	47.4
H7	143.7	82.0	42.9	368.0	185.2	49.7	586.3	306.4	47.7	981.7	498.9	49.2	47.4
H8	143.7	81.6	43.2	368.0	178.7	51.4	586.3	308.3	47.4	981.7	485.0	50.6	48.2
H9	176.3	85.2	51.7	402.8	191.4	52.5	735.8	331.8	54.9	1073.5	513.2	52.2	52.8
H10	177.2	83.7	52.8	411.4	197.8	51.9	721.8	342.8	52.5	1114.2	541.7	51.4	52.1

Note: *CT_o = Initial assembly cycle time (unit time)* *CT_{avg} = AverageCT (unit time)*
I = CT's Improvement=$(CT_o-CT_{avg})*100/ CT_o$

The results in table 6.4 show that the heuristics H1, H2, H5, H6, H7, H8, H9 and H10 can produce almost the same quality of the final schedule. The performance of these heuristics is almost the same between a simple problem (dataset 2) and a complex problem (dataset 5). In this experiment, an ordered constructive heuristic always produces a superior initial solution compared to a random constructive heuristic. A heuristic that uses an ordered constructive heuristic (H2, H6, H7 and H8) usually obtains a slightly better final schedule compared to heuristics, which start with a random constructive heuristic (H1, H5, H9 and H10). Therefore, it is worth using an ordered constructive heuristic rather than a random constructive heuristic since the time taken to produce the initial schedule is almost the same (i.e. about 0.06 seconds) although this is not generally true across all problems. Furthermore, by using an ordered constructive heuristic we can produce the first PCB faster than if we use a random constructive heuristic. In general, with the exception of H3 and H4, we improve the initial cycle time by 47.4% to 52.8%. H3 and H4 (that accept all moves) only obtained an improvement of 21.2% and 19.3% over the initial CT. The machine can start the assembly operation almost immediately after the PCB input data has been downloaded onto the machine. For example, we take less than 100 milliseconds to produce an initial schedule for 200 assembly points with 40 component types. While the machine is moving, picking up and placing components for the first PCB, our proposed on-line scheduler (that proposed in chapter 5) could employ the CPU free time to improve the initial schedule such that the pick-and-place process for subsequent PCBs will use an improved schedule. For example, in this experiment we take less than 100 seconds to make a CT improvement between 47.4% and 52.8% over the initial CT. The improvement process will continue since it does not incur 'significant cost' and perhaps we may obtain a very good quality solution (if the SMD placement machine produces a large number of PCBs) as our random greedy search heuristic approach is capable of avoiding local optimum.

Figure 6.3 shows the performance of each heuristic across various problem sizes (datasets). In this graph (figure 6.3), we plotted the CT's average of ten runs for each datasets. Apparently, the improvement of the heuristics that use a random constructive heuristic (i.e. H1, H5, H9 and H10 with the exception of

H3) show a slightly better CT's improvement over the initial solutions compared to the heuristic that use an ordered constructive heuristic (i.e. H2, H6, H7 and H8). However, if we consider the absolute performance rather than the improvement achieved (CT_{avg} in table 6.4), we can conclude that, in general, the ordered constructive heuristics (i.e. H2, H6, H7 and H8) produce (on average) a slightly better final solution quality than the random constructive heuristics (i.e. H1, H5, H9 and H10).

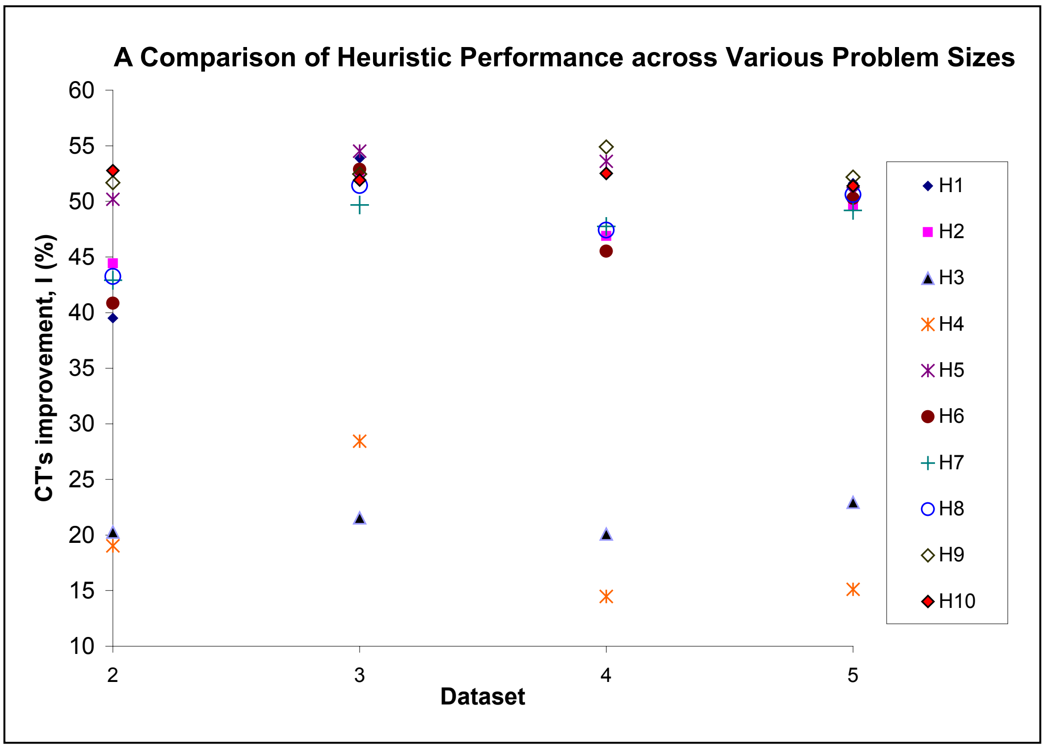


Figure 6.3: A comparison of heuristic performance across various problem sizes.

6.3.3 Discussion

We have developed two constructive heuristics (randomised and ordered constructive heuristics) to generate an initial solution. We apply three local search methods in determining the move, these being a random descent, a random move and a steepest descent. A random greedy search heuristic, which works at the higher level, will randomly choose a low-level heuristic in each iteration. Results show that with the exception of H3 and H4, we improve the initial cycle time by 47.4% to 52.8%. H3 and H4 (that accept all moves) only obtained an improvement of 21.2% and 19.3% over the initial CT. Results also indicate that a greedy search heuristic is suitable for solving the component pick-and-place problem on multi-head SMD placement machine and produces a good quality schedule. The on-line scheduling approach as proposed in chapter 5, can continually produce improved schedules without incurring any cost. As a result, applying on-line scheduling with a random greedy search heuristic to optimise the components' pickup-and-placement operation on multi-head SMD placement machine can increase the production throughput. Moreover, this can be achieved without paying significant extra cost.

As our simple greedy search hyper-heuristic has shown fairly good performance, we now extend this work by introducing a Monte Carlo hyper-heuristic in the following section.

6.4 Monte Carlo Hyper-heuristic

In this section we introduce a Monte Carlo based hyper-heuristic. The Monte Carlo hyper-heuristic manages a set of low-level heuristics (in this case just simple 2-opt swaps but they could be any other heuristics). Each of the low-level heuristics is responsible for creating a unique neighbour that may be impossible to create by the other low-level heuristics. At each iteration, the Monte Carlo hyper-heuristic randomly calls a low-level heuristic. The new solution returned by the low-level heuristic will be accepted based on the Monte Carlo acceptance criteria. The Monte Carlo acceptance criteria always accepts an improved solution. Worse solutions will be accepted with a certain probability, which decreases as the solutions worsen, in order to escape local minima. We develop three hyper-heuristics based on a Monte Carlo method, these being Linear Monte Carlo (LMC), Exponential Monte Carlo (EMC) and Exponential Monte Carlo with counter (EMCQ). We also investigate four other hyper-heuristics to examine their performance and for comparative purposes. To demonstrate our approach we employ these hyper-heuristics to optimise component pick-and-place sequencing in order to improve the efficiency of the multi-head SMD placement machine.

6.4.1 Monte Carlo Algorithm

Let us define a solution space *S*, an objective function *f* and a neighbourhood structure *n*. A basic Monte Carlo (MC) method for minimisation problem can be expressed by the following algorithm in figure 6.3 (Glover and Laguna, 1995):

```
Step 1: (Initialisation)
    (A) Choose a starting solution $S_0 \in S$;
    (B) Record the best obtained solution, $S_{best} = S_0$ and $f(S_{best}) = f(S_0)$;

Step 2: (Choice and termination)
    (A) Choose $S_c \in n(S_0)$;
    (B) Compute $\delta = f(S_c) - f(S_0)$;
    (C) If $\delta \le 0$ then accept $S_c$ (and proceed to Step 3);
    (D) Else: Accept $S_c$ with a probability that decreases with increases in
        $\delta$. If $S_c$ is rejected and stopping condition=false, then return to
        Step2(A);
    (E) Terminate by a stopping condition.

Step 3: (Update)
    Re-set $S_0 = S_c$, and if $f(S_c) < f(S_{best})$, return to Step1(B). Return to Step2
    if stopping condition=false.
```

Figure 6.4: A basic Monte Carlo (MC) algorithm.

In this work, we develop three types of acceptance criteria (referring to Step 2(D) figure 6.3):

a) Linear Monte Carlo (LMC). The probability is computed by *$(M-\delta)$* where *M* is a constant valued between 0 and 100. Based on our preliminary testing, LMC approach works well with *$M=5$* (for our test data, different values of *M* may be required for different problem instances). The test shows that the LMC approach is parameter sensitive. LMC approach will perform almost similar to a steepest descent approach with a small value of M since the probability of accepting a worse solution is too small. On contrary, larger *M* is more likely to lead to the acceptance of worse solution and it cannot converge. The new solution, S_c is accepted if a generated random number is less than $(M-\delta)$.

b) Exponential Monte Carlo (EMC). The probability is computed by $e^{-\delta}$ where $\delta=f(S_c)-f(S_0)$. The probability of accepting a worse solution decreases as δ increases. The new solution, S_c is accepted if a generated random number is less than $e^{-\delta}$.

c) Exponential Monte Carlo with counter (EMCQ). The probability is computed by $e^{-\theta/\tau}$ where $\theta=\delta*t$ and $\tau=\rho(Q)$. t is a computation time (in our case we use minutes as a unit time). θ and τ are defined such that we ensure that the probability of accepting a worse solution decreases as the time increases and δ increases. The factor of time is included in this formulation as an intensification factor. At the beginning of the search, the moderately worse solution is more likely to be accepted but as the time increases the worse solution is unlikely to be accepted. However, the probability of accepting a worse solution increases as the counter of consecutive non improvement iterations, Q increases. This is a diversification factor. $\rho(Q)$ is a function to intelligently control the Q. In this work we use $\tau=v*Q$ where $0\leq v\leq 1$, in order to limit the acceptance probability. However, our preliminary experiment on parameter sensitivity of v shows that the EMCQ approach with $v=1$ performs the best. In fact, the EMCQ is not sensitive to the value of v (i.e. any value of v produces almost the same quality of result). Therefore, we set $\tau=Q$. The new solution, S_c is accepted if a generated random number is less than $e^{-\theta/\tau}$.

The formulation of the acceptance criteria of EMC and EMCQ approach is quite similar to the acceptance criteria of a simulated annealing approach. The difference, for the EMC and EMCQ approach is that we do not have a cooling schedule. EMCQ approach will exponentially increase the acceptance probability as we have been unable to find a better solution for a long time (i.e. too long being trapped in local optima). However, EMCQ approach will exponentially reduce the acceptance probability as the searching time increases (similar to a simulated annealing approach). As EMC and EMCQ approaches do not have parameters, which have to be carefully tuned (all the parameters are automatically controlled based on the solution quality (with the exploration time and the duration of being trapped in local optima in the case of EMCQ)), these methods are simple and robust heuristic technique.

The EMC and EMCQ approaches work as follows. First, an initial solution is chosen. Then, for each iteration, a neighbour of the current solution is generated. The 'qualities' of the two solutions are compared. A decision is made whether the new solution should be accepted. An improved solution is always accepted. However, in order to escape from the local optima we accept a worse solution with a probability that depends on δ (and the duration we have been trapped in the local optima in the case of EMCQ approach). A worse solution is more likely to be accepted if the δ is small (and we cannot find a better solution for a long time for the case of EMCQ approach). The idea of EMCQ approach is to ensure that we only accept a moderately worse solution after most of the neighbours of the current solution have been explored and none of them is better than the old solution. Table 6.5 shows the difference among the three proposed acceptance criteria.

TABLE 6.5: A COMPARISON OF ACCEPTANCE PROBABILITY FOR LMC, EMC AND EMCQ

Method	**Acceptance Probability**
LMC	Accept S_c if $\delta \le 0$, otherwise accept S_c if $x<(M-\delta)$.
EMC	Accept S_c if $\delta \le 0$, otherwise accept S_c if $x<e^{-\delta}$ where $\delta=F(Sc) - F(S0)$.
EMCQ	Accept S_c if $\delta \le 0$, otherwise accept S_c if $x<e^{-\theta/\tau}$ where $\theta=(\delta * t)$ and $\tau=Q$.

Note: x = generated random number; *S_c = a trial solution.*

6.4.2 Hyper-Heuristics for Multi-Head SMD Placement Machine

In this work, we investigate seven hyper-heuristic approaches with different acceptance criteria (see 2(D), figure 6.3):

a) AM (All Move): Randomly select LLH and accept any solution returned by the LLH.

b) OI (Only Improving): Randomly select LLH and only accept an improved solution returned by the LLH.

c) OICF (Only Improving Choice Function): Select LLH based on historical performance (Cowling et al., 2001a) and only accept an improved solution returned by the LLH.

$F(N_k)=max\{\alpha*f_1(N_k)+ \beta*f_2(N_j,N_k)+ \sigma*f_3(N_k)\}$

Where

$F(N_k)$ is a Choice Function of the k^{th} LLH that has the largest $F(N_{k.})$. $f_1(N_k)$ is the cumulative performance rate of heuristic N_k, $f_2(N_j,N_k)$ is the cumulative performance rate of consecutive pairs of heuristics (heuristic N_j followed by N_k) and $f_3(N_k)$ is the CPU time, which has elapsed since heuristic N_k was last called. Details of the algorithm can be found in (Cowling et al., 2001a, b). If the time taken by each LLH to make a swap is too short (approximate to zero millisecond), then we set the duration as 1.

d) AMCF (All Move Choice Function): Same as OICF but in this case we accept any solution returned by the LLH.

e) LMC (Linear Monte Carlo): Randomly select LLH and accept S_c returned by the LLH based on the LMC acceptance criteria as in table 6.5.

f) EMC (Exponential Monte Carlo): Randomly select LLH and accept S_c returned by the LLH based on the EMC acceptance criteria as in table 6.5.

g) EMCQ (Exponential Monte Carlo with Counter): Randomly select LLH and accept S_c returned by the LLH based on the EMCQ acceptance criteria as in table 6.5.

Table 6.6 summarises the differences across these hyper-heuristics approaches.

TABLE 6.6: A LIST OF HYPER-HEURISTICS WITH THEIR ACCEPTANCE CRITERIA

Hyper-heuristic	**Accepting Criteria**
AM	Accept all moves.
OI	Accept S_c if $\delta \leq 0$, otherwise reject S_c.
OICF	Accept S_c if $\delta \leq 0$, otherwise reject S_c.
AMCF	Accept all moves.
LMC	Accept S_c if $\delta \leq 0$, otherwise accept S_c if $x<(M-\delta)$.
EMC	Accept S_c if $\delta \leq 0$, otherwise accept S_c if $x<e^{-\delta}$ where $\delta = F(Sc) - F(S0)$.
EMCQ	Accept S_c if $\delta \leq 0$, otherwise accept S_c if $x<e^{-\theta/\tau}$ where $\theta=(\delta * t)$ and $\tau=Q$.

6.4.3 Low-Level Heuristics

The low-level heuristics are implemented based on the problem domain. A set of simple LLHs provides more flexibility for the hyper-heuristic. A set of complex LLH, such as steepest descent that finds the best neighbour is computationally expensive and indirectly influences the hyper-heuristic to behave as a steepest descent method (for example). This results in simple hyper-heuristics (such as AM) and more complex hyper-heuristics (such as EMC and EMCQ) producing similar results when using a complex set of LLH. This is due to the fact that the complex set of LLH is able to find good solutions without having to be guided by a hyper-heuristic. Of course, it takes a lot longer to implement complex LLH's when the problem domain changes. Therefore, if we are dealing with a set of complex LLHs, it may be worth applying a simple hyper-heuristic such as an AM hyper-heuristic. However, in this work we prefer to use a set of simple LLHs with an intelligent hyper-heuristic as this allows us to solve a wider range of problems.

Our LLH is a set of 2-opt operations. Each LLH is responsible for creating a unique neighbour that may be impossible to reach by other LLHs. In fact, each LLH plays a unique role in minimised the cycle time. For example, one aims to minimise the pickup sequence, whilst others aim to minimise the placement sequence, the pickup nozzle assignment, the sub-tour's grouping etc. Whatever factor is being minimised the overall aim is to minimise CT. There is no good strategy for selecting which LLH to apply at a given time. However, continuously applying single LLH will quickly lead to a local optima.

The aim of optimising the pickups and the placements is to minimise the CT. However, there is a trade-off between optimising the pickups and optimising the placements. As mentioned earlier, applying the hyper-heuristic over of a set of LLHs simplifies the problem such that we do not have to compromise between optimisation of picking and placing.

On the contrary, in solving this problem, a typical meta-heuristic approach has to intelligently deal with the trade-off between optimising the picking and placement. A meta-heuristic approach is also forced to make a decision as to which sequence of factors need to be minimised and how far each factor should be minimised. Many researchers simplified the problem and solved the component pick-and-place sequencing of the multi-head SMD placement by ignoring these factors.

However, the problem can be simplified by applying a hyper-heuristic over a set of LLHs. In this work, we develop six simple LLHs. These LLHs are very similar to the LLHs in chapter 5 (see section 5.6), except that, in this work we only choose one neighbour solution at each iteration, whereas in chapter 5, we explore *G* neighbour solutions at each iteration.

6.4.4 Testing and Results

An initial solution is generated using either the randomised or the ordered constructive heuristic that we proposed in section 5.3.1. We use the same assumptions as in chapter 5, since we model the same placement machine as in that chapter. We also apply the same experimental parameters.

In this experiment we use two datasets (dataset N80K20_A and N240K40_F). Dataset N80K20_A (see appendix A) has 80 PCB points (N) consisting of 20 component types (K) with board width, BW=200 and board length, BL=600. Dataset N240K40_F (see appendix A) has N=240, K=40, BW=600 and BL=1800. These datasets are randomly generated using our random PCB generator software called PCBgen. PCBgen allows the user to set the required N, K, BW and BL. The PCBgen software is available at http://www.cs.nott.ac.uk/~gxk/.

We ran the experiments using an Intel® Pentium®4 PC with a 1.5GHz processor and 256 MB RAM. In this work we set *M=5* (for LMC), *α=1.0*, *β=0.01* and *σ=0.5* (for OICF and AMCF). The parameter values are chosen based on the best result obtained from our preliminary testing on parameter sensitivity. The parameter sensitivity tests showed that LMC, OICF and AMCF hyper-heuristics performance are very sensitive to their parameters and the best value for the parameters is subject to the problem size. Table 6.7 shows the experimental results of an average of ten runs on dataset N80K20 and N240K40 with each run being given one hour of computation time as a termination criteria. However, any other termination criteria is also applicable. For example, if we apply these methods for the on-line scheduling approach that we proposed in chapter 5, we can continually search for an improved schedule until there are no more PCB's to be assembled. The initial CT for dataset N80K20_A and N240K40_F is 1061.04 and 8385.98 unit time, respectively. For a fair comparison, all approaches use the same initial solution. The figures in table 6.7 show the average of the best obtained solution's qualities with the computation time (when the best solution is found).

TABLE 6.7: AN AVERAGE RESULT OF TEN RUNS ON EACH DATASET (TEST DURATION: 1 HOUR)

	Dataset N80K20_A (CT_o=1061.04)			**Dataset N240K40_F (CT_o=8385.98)**		
	CT_{avg}	**T**	**I(%)**	**CT_{avg}**	**T**	**I(%)**
AM	720.15	31.43	32.13	7602.68	25.76	9.34
LMC	266.80	50.20	74.85	2350.20	58.38	71.97
EMC	281.84	26.96	73.44	2350.62	55.84	71.97
EMCQ	248.80	45.80	76.55	2370.30	59.22	71.73
OICF	342.98	10.20	67.68	2968.45	27.00	64.60
AMCF	804.30	18.20	24.20	8116.27	5.31	3.22
OI	288.98	54.39	72.76	2771.10	57.72	66.96

Note: CT_{avg}=Average CT (time); T=computation time (minutes;)
CT_o=Initial CT;
*I = CT's Improvement=(CT_o-CT_{avg})*100/ CT_o.*

The results in table 6.7 show that the OI hyper-heuristic, that only accepts an improved solution (i.e. typical descent method), rapidly converges to a local optima. For example, in dataset N80K20_A, the OI hyper-heuristic gets trapped in local optima in 2.58 minutes and cannot find a better solution even after one hour. However, for the larger size dataset, N240K40_F, the OI hyper-heuristic can still improve the solution (not yet in local optima). In all tests, the OICF (Cowling et al., 2001a, b) hyper-heuristic does not perform well, being even worse than OI hyper-heuristic for dataset N80K20_A. This indicates that the historical performance and the time taken by the LLH to produce a neighbour solution are not applicable in this case study. This is because the performance of the LLH is unpredictable since it generates a random neighbour every time it is called and historical performance counts for nothing. In fact, the LMC, OI, EMC and EMCQ hyper-heuristics that randomly call the LLH are superior to OICF and AMCF, with the EMCQ hyper-heuristics being superior to the other hyper-heuristics (for dataset N80K20_A). This indicates that the formulation of an exponential acceptance criterion is effective in searching for better solutions. The exponential formulation produces a lower probability of acceptance for higher search times and worse evaluation, such that it is adequate to guide the direction of the hyper-heuristic. Injecting a counter of consecutive unimproved solutions into the EMCQ formulation gives a significant impact in guiding the search direction and it also provides a way to diversify the search when trapped in a local optima.

Figure 6.5 shows the distribution of CT's improvements obtained in different hyper-heuristic approaches (using the results of ten runs on dataset N80K20_A for a duration of 1 hour). From figure 6.5 it can be seen that the EMCQ hyper-heuristic has outperformed the other hyper-heuristics. It shows a fairly consistent performance with only a small variation of improvement (for dataset N80K20_A).

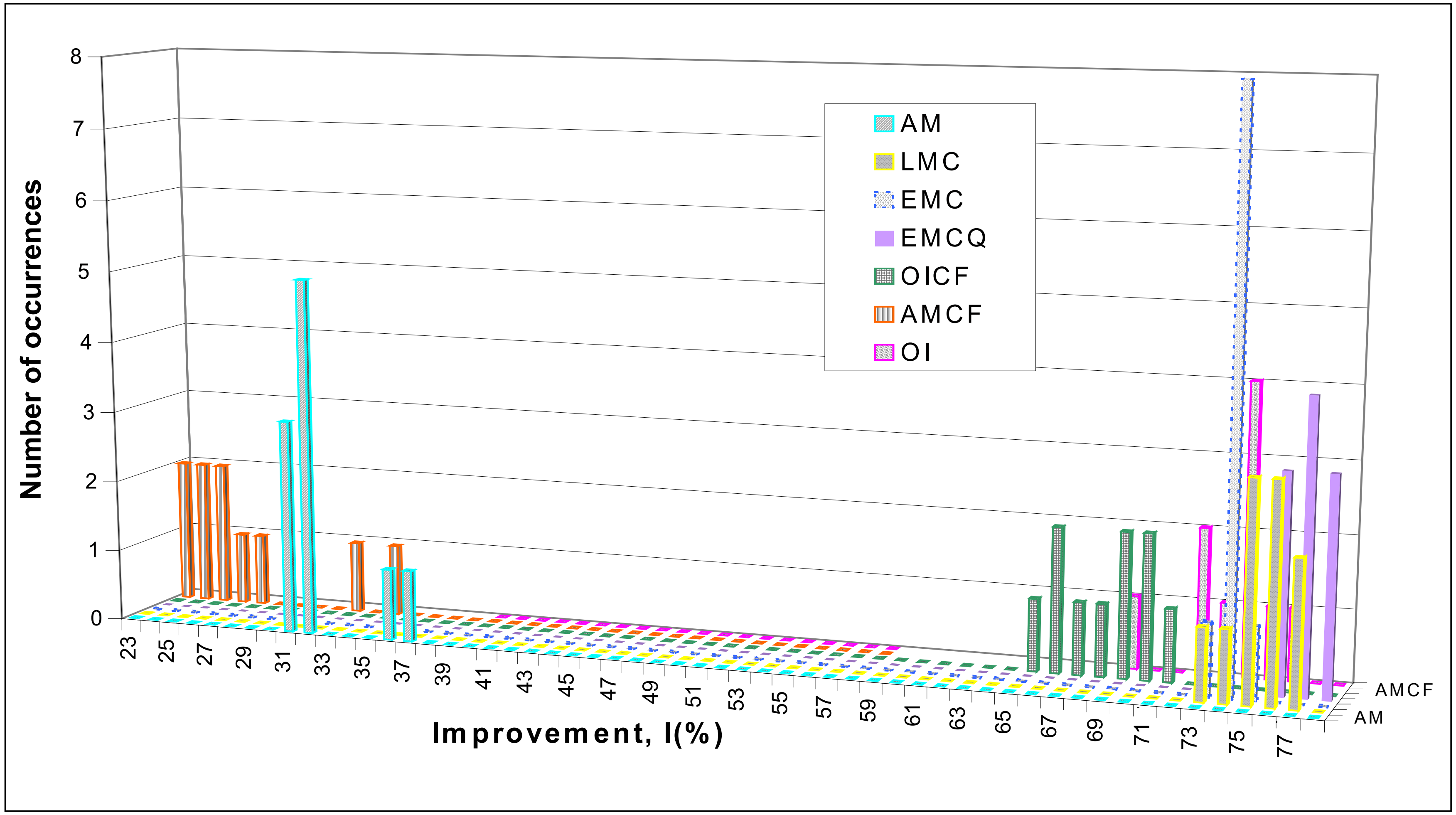


Figure 6.5: Distribution of CT's improvement, I obtained in different hyper-heuristic approaches on dataset N80K20_A (test duration: 1 hour).

This work is actually a continuation of our work in chapter 5, where we employ on-line scheduling to continually search for an improved solution, while the placement machine is picking up and placing components onto the PCB. These approaches require a fast searching technique that is capable of finding good quality solutions in short timescales. Thus, we examine the performance of the hyper-heuristics after five minutes to identify a good hyper-heuristic technique that is able to operate over a shorter timescale. These results are shown in table 6.8, which demonstrate that the LMC, EMC and EMCQ hyper-heuristics also perform well in short timescales. These results demonstrate that the EMCQ hyper-heuristic is a good and fast hyper-heuristic approach as well as operating well over longer time periods (table 6.7). The behavior of the hyper-heuristics over time can be observed in figure 6.6.

TABLE 6.8 AN AVERAGE RESULT OF TEN RUNS ON EACH DATASET (TEST DURATION: 5 MINUTES)

	Dataset N80K20_A (CT_o=1061.04)			**Dataset N240K40_F (CT_o=8385.98)**		
	CT_{avg}	**T**	**I(%)**	**CT_{avg}**	**T**	**I(%)**
AM	750.83	5.00	29.24	8873.95	5.00	-5.82
LMC	317.20	5.00	70.10	3169.99	5.00	62.20
EMC	332.54	5.00	68.66	3291.70	5.00	60.75
EMCQ	307.20	5.00	71.05	3197.70	5.00	61.87
OICF	347.67	5.00	67.23	4279.74	5.00	48.97
AMCF	862.50	9.60	18.71	8269.00	5.00	1.39
OI	322.63	5.00	69.59	3346.14	5.00	60.10

Figure 6.6 shows that the AMCF hyper-heuristic is the worst among the other hyper-heuristics. This indicates that the heuristic's selection based on the Choice Function criteria does not perform well in this case study. The OI and OICF hyper-heuristics get trapped in local optima whilst the LMC, EMC and EMCQ hyper-heuristics continually find improved solutions. As the performance of the OI and OICF hyper-heuristics are almost the same in this case study, we can conclude that the heuristic's selection in OICF is arbitrary.

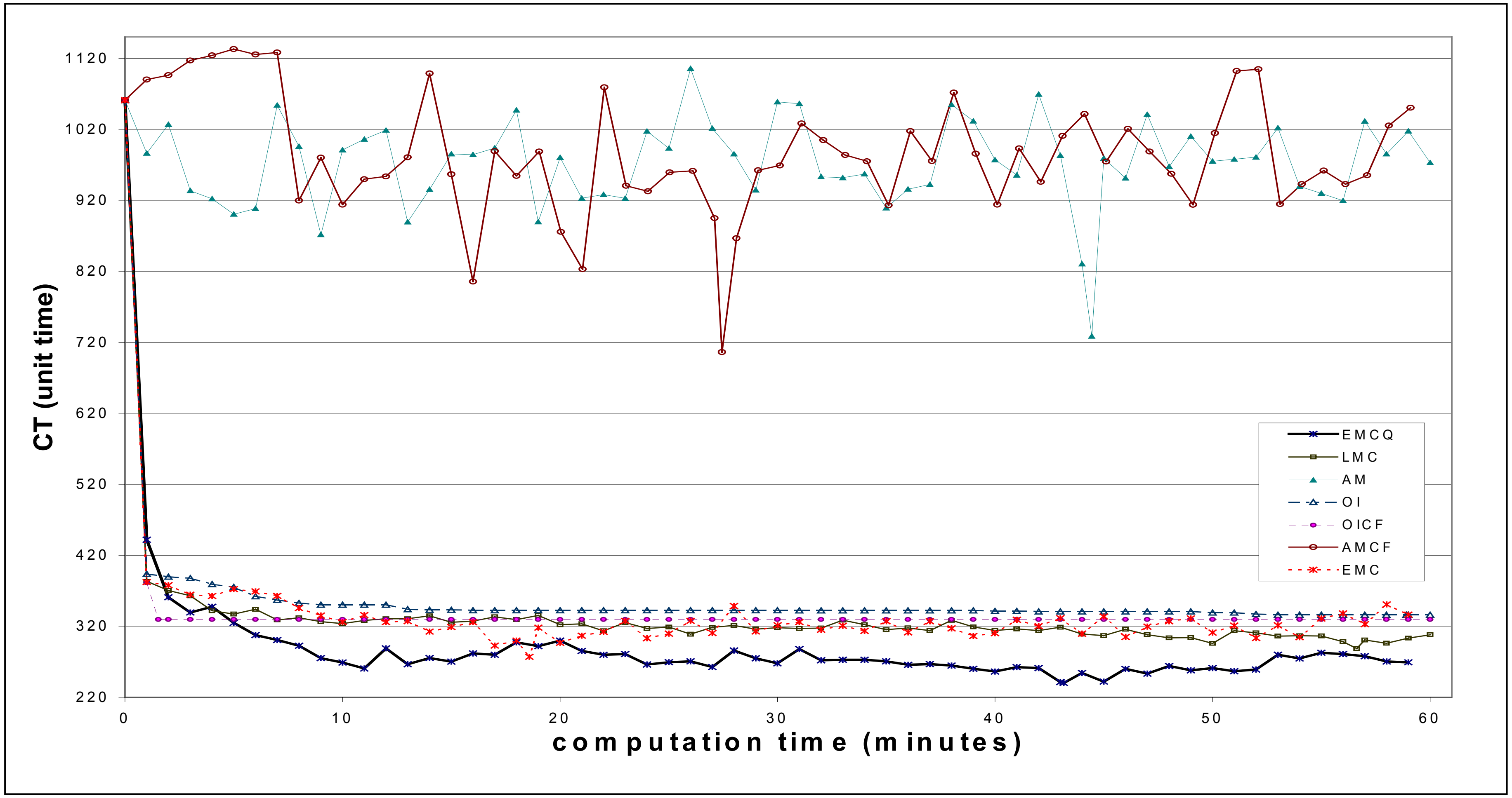


Figure 6.6: A comparison of hyper-heuristics behaviour.

6.4.5 Discussion

We have examined seven hyper-heuristic approaches, these being AM, OI, OICF, AMCF, LMC, EMC and EMCQ. The OICF and AMCF are the hyper-heuristics that choose the next LLH to be applied based on previous performance, whilst the other hyper-heuristics randomly select the next LLH to be called. Since all the LLHs produce a random neighbour solution each time they are called, their performance is unpredictable. Thus, hyper-heuristics based on previous LLH's performance (OICF and AMCF) are unable to perform well. On the other hand, the other hyper-heuristics demonstrated a good performance (except AM), especially LMC and EMCQ. For example, for dataset N80K20_A, the EMCQ hyper-heuristic minimised the CT by 76.55% respectively (with respect to the initial solution). Generally, LMC and EMCQ hyper-heuristics show almost equal performance in this case study, but the EMCQ hyper-heuristic has a better formulation, which include the intensification (time, t) and diversification (counter for consecutive unimproved, Q) factors. Moreover, the EMCQ hyper-heuristic is not parameter sensitive (for this problem) heuristic whereas LMC is sensitive to its parameter M. Therefore, the EMCQ hyper-heuristic is a good, fast and robust hyper-heuristic.

6.5 Summary

Since the component pick-and-place sequencing problem of multi-head SMD (surface mount device) placement machine has various optimisation factors, such as optimisation of the pickup sequence, the placement sequence, pipette/nozzle assignment, sub-tour grouping and sequencing the sub-tour, it is difficult to devise a good strategy to minimise the assembly cycle time. However, by applying a hyper-heuristic approach, we do not have to concern ourselves with the trade-off between the optimisation of the important factors as this will be catered for within the hyper-heuristic. We have demonstrated the ability of hyper-heuristic approaches in solving the component pick-and-place sequencing problem of multi-head SMD placement machine. In section 6.3, we presented a greedy search hyper-heuristic that randomly calls low-level heuristics (LLHs) and accepts any returned solution. The greedy search hyper-heuristic operates with six LLHs. Instead of using a simple descent method in each local search (i.e. LLH), as used in chapter 5, we apply three neighbourhood search techniques to determine a move from one schedule to another. These being random descent, random move and steepest descent acceptance criteria.

To further investigate the effectiveness of hyper-heuristic approaches in solving the component pick-and-place sequencing problem, we then proposed a Monte Carlo based hyper-heuristic. In this case study, the Monte Carlo hyper-heuristic randomly calls a LLH. However, one can intelligently choose the LLH to be applied. The new solution returned by the low-level heuristic will be accepted based on the Monte Carlo acceptance criteria. The Monte Carlo acceptance criteria always accept an improved solution. Worse solutions will be accepted with a certain probability, which decreases with worse solutions, in order to escape local minima. We developed three hyper-heuristics based on a Monte Carlo method, these being Linear Monte Carlo (LMC), Exponential Monte Carlo (EMC) and Exponential Monte Carlo with counter (EMCQ). Experimental results show that, in general, LMC and EMCQ hyper-heuristics show almost equal performance in this case study, but the EMCQ hyper-heuristic has a better formulation, which includes the intensification (time, t) and diversification (counter for consecutive unimproved, Q) factors. Moreover, the EMCQ hyper-heuristic is 'not a parameter sensitive' (for this problem) hyper-heuristic whereas LMC is sensitive to the parameter M. Therefore, the EMCQ is a good, fast and robust hyper-heuristic. The proposed EMC and EMCQ acceptance criteria could be applicable on solving other machine problems. Indeed, it could be applicable on solving other NP hard problems. For example, the EMC and EMCQ acceptance criteria have been successfully adopted by Abdullah et al. (2004) and Kendall and Mohamad (2004a) to solve the examination timetabling and frequency assignment problems, respectively. However, some modifications such as on parameters $v=1$and t calculated in minutes as a unit time, might be required to suit the problem specific constraints. The EMCQ hyper-heuristic could also be applicable on solving other NP hard problems, which might require some modifications as discussed above. Indeed, the low-level heuristics are problem specific.

Since the problem requires various neighbourhood structures, a Variable Neighbourhood Search might be suitable for solving this type of problem. Thus, in the next chapter, we employ a Variable Neighbourhood Search approach to solve the component pick-and-place sequencing problem of multi-head SMD placement machine.

Chapter 7

A Variable Neighbourhood Monte Carlo Search For Multi-Head Surface Mount Device Placement Machine

7.1 Introduction

In the previous chapter we showed that our proposed theoretical methodology for on-line scheduling and various hyper-heuristic approaches was suitable to solve the component pick-and-place operation of a theoretical multi-head surface mount device (SMD) placement machine. Hence, this chapter extends the study to apply a VNS (Variable Neighbourhood Search) approach for optimising the component pick-and-place sequencing problem of a multi-head SMD placement machine. We develop a Variable Neighbourhood Monte Carlo Search (VNMS), which employs a variable neighbourhood search technique with an Exponential Monte Carlo acceptance criterion. The motivation for investigating a variable neighbourhood search (VNS) approach is that the VNS framework provides a way of exploring various neighborhood structures in solving a problem and none of the reported work has attempted to apply variable neighbourhood search to optimise the SMD placement machine. As the component pick-and-place sequencing problem involves a number of intertwined scheduling problems that result in various neighbourhood structures, the VNS approach seems to be appropriate for solving this type of problem. The hyper-heuristic approach might also suitable for solving this problem and has been studied in chapter 6. This suggests that a VNS approach might be suitable as a hyper-heuristic by allowing various neighbourhoods to be searched, which is the principle behind VNS.

VNMS is a descent-ascent heuristic that operates on three sets of neighbourhood structures that are based on three different local search operators. The first two sets use a steepest descent and Exponential Monte Carlo local search, respectively whilst the third set uses a random 3-opt operator. The solution returned by a local search after exploring a neighbourhood structure will be accepted based on the EMCQ (Exponential Monte Carlo with counter) acceptance criterion. The work presented in this chapter has been disseminated as follows:

> Ayob M. and Kendall G. (2005b). A Variable Neighbourhood Monte Carlo Search For Component Placement Sequencing Of Multi-head Placement Machine in Printed Circuit Board Assembly. Submitted to the *Intelligent Manufacturing Journal*.

7.2 Variable Neighbourhood Search Background

Variable neighbourhood search (VNS) is a relatively unexplored approach (Hansen and Mladenović, 2001), being introduced by the same authors in 1997 (Hansen and Mladenović, 1997). By systematically changing the neighbourhood within a local search algorithm, VNS can explore distant neighbourhoods of the current solution, and jump to a new solution if it is superior to the current solution (Hansen and Mladenović, 1997). A local search is applied repeatedly to obtain the local optima from the selected neighbouring solution. The basic VNS algorithm is a descent heuristic but it can be transformed to a descent-ascent or a best improvement method (Hansen and Mladenović, 2001). Let n_w, $w=1,2\ldots,W$, be a set of predefined neighbourhood structures, and $n_w(x)$ is the set of solutions in the w^{th} neighbourhood of x, $f(x)$ is the quality of solution x. W is the total number of neighbourhood structures to be used in the search. The basic VNS algorithm (Hansen and Mladenović, 1997) is presented in figure 7.1.

Recently, there has been increasing interest in the VNS approach. For example, Avanthay et al. (2003) developed an adaptation of VNS to solve the graph coloring problem with a Tabucol (a variant of tabu search) algorithm (Hertz and Werra, 1987) as a local search. They used three neighbourhood structures; these being vertex, class, and non-increasing neighbourhoods. Their VNS however is not superior to the hybrid algorithm proposed by Galinier and Hao (1999) that integrates a tabu search and a genetic algorithm. Fleszar and Hindi (2004) applied a VNS approach in solving the resource-constrained project scheduling problem. Results show that the quality of solutions and lower bounds achieved by VNS with enhanced moves and precedence augmentation is impressive. Other works on VNS can be found in (Caporossi et al., 1999; Caporossi and Hansen, 2000, 2004; Crainic et al., 2004; Hansen and Mladenović, 2001; Mladenović and Hansen, 1997; Mladenović et al., 2003a, 2003b; Morena Pérez, et al., 2003; Polacek et al., 2004), which demonstrate that it is suitable across a number of different problem types.

```
Initialisation: Select the set of neighbourhood structures $n_w$, $w=1,2\ldots W$,
                that will be used in the search; find an initial solution $x$;
                choose a stopping condition;

Main step:      Set $w \leftarrow 1$ until $w=W$, repeat the following steps until the
                stopping condition is met:
                  a). Shaking. Generate a point $x'$ at random point from the
                      $w^{th}$ neighbourhood of $x$ ($x' \in n_w(x)$);
                  b). Local search. Apply some local search method with $x'$ as
                      an initial solution to obtain the local optima, $x''$;
                  c). Acceptance criteria (Move or not). If $x''$ is accepted,
                      then $x \leftarrow x''$. If $x''$ is superior to the incumbent solution
                      $x$, then continue the search with $n_1(w \leftarrow 1)$; otherwise, set
                      $w \leftarrow w+1$;
```

Figure 7.1: The basic VNS algorithm (Hansen and Mladenović, 1997, 2001).

7.3 Variable Neighbourhood Search for Component Pick-and-Place Sequencing

Originally, the basic VNS approach proposed by Mladenović and Hansen (1997) was a descent heuristic. However, in this work we develop two types of VNS approaches, these being basic VNS (VNS-1, VNS-2 and VNS-3) and a Variable

Neighbourhood Monte Carlo Search (VNMS). The first one is based on the basic VNS (Mladenović and Hansen, 1997) that operates on different local searches. The second type of VNS (VNMS) has three stages where each stage operates on a different set of local searches.

7.3.1 Acceptance Criteria

In this work, we use three acceptance criteria, which are applied at the VNS level and/or the local search level. These are: a descent that only accepts an improved solution, EMC (Exponential Monte Carlo) and EMCQ (Exponential Monte Carlo with counter). The EMC and EMCQ acceptance criteria have been proposed in chapter 6 (see section 6.4.1).

7.3.2 Basic Variable Neighbourhood Search

Based on the basic VNS algorithm in figure 7.1 and the three acceptance criteria, we implement three heuristics, these being VNS-1, VNS-2 and VNS-3.

VNS-1 is absolutely a basic descent heuristic that operates with a random descent (RD-LS) local search by using descent acceptance criteria at the local search and VNS levels. VNS-1 only accepts an improved solution returned by the RD-LS local searches. The local search starts by randomly generating a solution *x'* from the w^{th} neighbourhood of *x* $(x' \in n_w(x))$. Starting from *x'* as an initial solution, the RD-LS sequentially visits at least [1]*P* neighbours around the w^{th} neighbourhood of *x'*. The RD-LS accepts the first improving neighbour solution and moves to this solution. Then the search continues until the stopping condition is true. We use two stopping conditions (in the local search), these being a number of visited neighbours and the solution's quality. If the current neighbour solution is better than the best obtained solution by the RD-LS, then the search continues even after exceeding the limit of visited neighbours. The algorithm for the RD-LS is shown in figure 7.2.

[1] $P = \sum_{e=1}^{G-1} e$ *where G is the number of pipettes/nozzles per head*

VNS-2 is also a basic descent heuristic but operates with EMC (Exponential Monte Carlo) local search (EMC-LS) where the acceptance criteria at the local search and VNS levels are different, these being EMC and descent acceptance criteria, respectively. VNS-2 itself is a descent heuristic but the EMC-LS local search is a descent-ascent heuristic. The EMC-LS also begins by randomly generating a solution *x'* from the w^{th} neighbourhood of *x* $(x' \in n_w(x))$. Starting from *x'* as an initial solution, EMC-LS sequentially visits *P* neighbours around the w^{th} neighbourhood of *x'*. The search sequentially moves to the new accepted solution even though the new solution is worse (depending on the EMC acceptance criteria). The EMC-LS exits when it reaches the limit of visited neighbours (*P*). It returns the best obtained solution to the VNS heuristic. Figure 7.2 also describes the EMC-LS algorithm.

```
(1). Begin the local search with a given solution, x from a neighbourhood
     structure $n_w$, set $x' \leftarrow x$.
(2). Randomly generate the index of the point to be visited, q where $q \in$
     $\{1,2,..,G\}$; Let us denote $b_q$ as the $b^{th}$ point that has $q^{th}$ indexing.
(3). Set $q \leftarrow 1$ until $q=G$, repeat the following steps:
     (a) Set $u \leftarrow (q+1)$ until $u=G$ (where u is a secondary index), repeat the
         following steps:
         (i)   Swaps the points $b_q$ and $b_u$ to produce $x''$ where $x'' \in n_w(x')$.
         (ii)  If first swapping operation, then
                  set $x' \leftarrow x''$, $f(x') \leftarrow f(x'')$, $x_{best}' \leftarrow x''$ and $f(x_{best}') \leftarrow f(x'')$.
               Otherwise;
                  evaluate the $f(x'')$.
                  If $f(x'')$ is accepted by the acceptance criteria, then
                     $x' \leftarrow x''$ and $f(x') \leftarrow f(x'')$.
                  If $f(x'') < f(x_{best}')$ then
                     $f(x_{best}') \leftarrow f(x'')$; $x_{best}' \leftarrow x''$.
                     Goto step 3a (for the RD case only)
         (iii) Set $u \leftarrow (u+1)$;
     (b) Set $q \leftarrow (q+1)$.
(4).    Return the best obtained solution, $x_{best}'$ and $f(x_{best}')$.
```

Figure 7.2: The RD-LS or EMC-LS local search for component pick-and-place sequencing of multi-head SMD placement machine used by VNS-1, VNS-2 and VNS-3 (minimisation problem).

VNS-3 is a basic VNS but it is a descent-ascent VNS with EMCQ acceptance criteria at the VNS level that operates with local searches that use an EMC acceptance criteria (EMC-LS). Instead of simple descent heuristics, the VNS-3 transforms the descent VNS into a descent-ascent VNS. It applies the EMCQ acceptance criterion (see section 6.4.1) to probabilistically accept a worse solution returned by the local searches. VNS-3 uses the same EMC local search, EMC-LS as in (VNS-2).

The shaking procedure in the basic VNS approach provides a diversification factor in order to reach a distant neighbourhood whilst the local search will intensify the search to make it converge to a local optimum. In this work the shaking procedure and the local search are integrated in one procedure for the case of VNS-1, VNS-2 and VNS-3.

Table 7.2 (in section 7.4) shows the difference among various hyper-heuristics and VNS approaches that are investigated in this work.

7.3.3 Local Search for Basic Variable Neighbourhood Search

A local search heuristic explores a single neighbourhood solution space by performing a sequence of local changes to an initial solution, in order to improve the quality of a solution until a local optima is found (Fleszar and Hindi, 2002). A neighbouring solution is produced by a move-generation mechanism and will be selected depending on the pre-defined acceptance criteria (Osman, 1996).

We develop two local searches, these being Random Descent (RD-LS) and Exponential Monte Carlo (EMC-LS) local searches. The RD-LS is a descent heuristic that only accepts an improved solution (it uses a descent acceptance criterion) whereas the EMC-LS is a descent-ascent heuristic that uses an EMC acceptance criterion. In these local search methods, we include the shaking procedure in the local search procedure (combining step *a* and *b* in figure 7.1). The resulting effect is similar to the basic VNS since the local search starts with the shaking procedure by accepting the first swaps in the local search. We use a 2-opt operator in the local search.

As the optimisation of the component placement sequencing of a multi-head placement machine deals with a number of sub tours, we need to determine which sub tour is to be explored at any given time. Hence, these methods randomly choose the available sub tour at each time the local search is called. On the other hand, the VNMS (Variable Neighbourhood Monte Carlo Search) approach consecutively explores all the sub tours each time the local search is called.

To explain the RD-LS and the EMC-LS, let us introduce an index *q*. The q^{th} index is randomly generated to avoid visiting consecutive points in the sequence, one after another, that may cause very deterministic moves and a cycling problem. We use indexing to ensure most of the neighbours will be visited. Moreover, having the index of visited points provides a systematic way of exploring the neighbours in the neighbourhood. Let us denote b_q as the b^{th} point that has q^{th} indexing. A move is generated as shown in figure 7.2.

The local search begins with an *x* solution that will be explored in w^{th} neighbourhood structure (determined by VNS at the higher level). Before beginning the search, *q's* index is randomly generated. Each point in a selected sub tour (or each sub tour) will randomly be associated with the *q's* index. The example of *q's* indexing procedure is explained in figure 7.3.

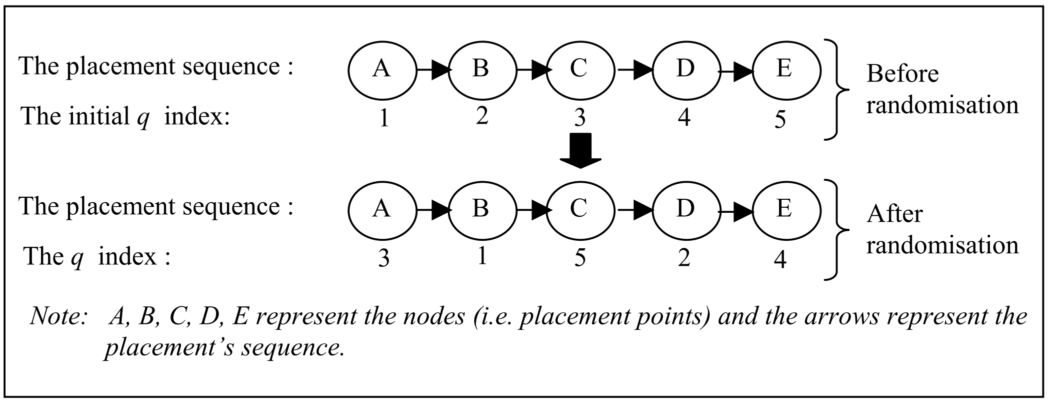


Figure 7.3: The example of *q* indexing procedure for the placement sequence.

Referring to figure 7.3, the *q's* index determines the sequence of points to be swapped in the 2-opt operation. For example, after a randomisation process, we obtained *q's* index as shown in figure 7.3. This means that the first neighbour solution to be generated is "*A→D→C→B→E*" as the placement points having the index *q=1* and *q=2* are swapped. The solution obtained will be used as an incumbent solution, x' for the subsequent swapping operations. The second swapping operation (points *q=1* and *q=3*) will produce a neighbour solution "*D→A→C→B→E*". If the solution is accepted (based on the acceptance criterion, then *x'* and *f(x')* are updated and the next neighbour solution to be generated will be "*D→E→C→B→A*". This process continues until the stopping condition is met.

The optimisation of component pick-and-place sequencing problem of a multi-head SMD placement machine involves optimisation within a sub tour and an optimisation among sub tours (assigning PCB points to a sub tour). Each sub problem's optimisation procedure explores a different solution space (neighbourhood). In this work we have developed six neighbourhood structures for the VNS-1, VNS-2 and VNS-3:

- n_1: The sub tour's neighbourhood, which changes the assignment of PCB points to a sub tour by applying the 2-opt operator.
- n_2: The pickup pipette/nozzle's neighbourhood, which changes the assignment of pickup pipettes/nozzles by applying the 2-opt operator.
- n_3: The pickup sequence's neighbourhood, which changes the pickup sequence by applying the 2-opt operator.
- n_4: The placement pipette/nozzle's neighbourhood, which changes the assignment of the placement pipettes/nozzles by applying the 2-opt operator.
- n_5: The placement sequence's neighbourhood, which changes the placement sequence by applying the 2-opt operator.
- n_6: The sub tour sequence's neighbourhood, which changes the sequence's order of sub tours by applying the 2-opt operator.

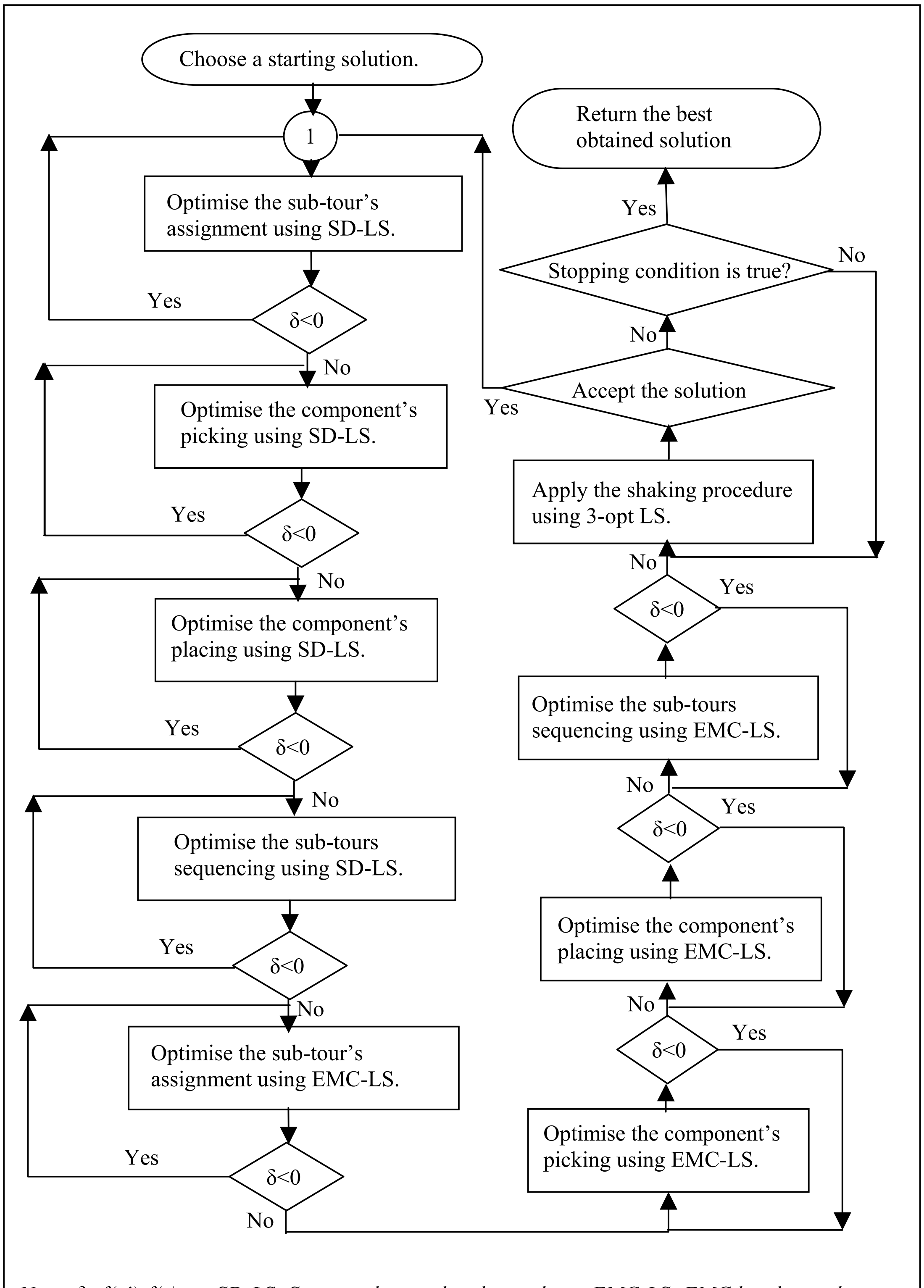
The neighbourhood is explored starting with the first neighbourhood, that is the sub tour's neighbourhood (n_1). Then, consecutively followed by the pickup pipette/nozzle (n_2), pickup sequence (n_3), placement pipette/nozzle (n_4), placement sequence (n_5) and sub tour sequence (n_6) neighbourhoods. The sequence of exploring the neighbourhood structure is determined based on the observation of our preliminary experiments. We usually obtained a large improvement if we explore the sub tour's neighbourhood, especially early in the search. After exploring the pickup pipette/nozzle neighbourhood, it is worth exploring the pickup sequence's neighbourhood since the cost of pickup sequence is dependent on the pickup pipette/nozzle assignment. Similarly, after exploring the placement pipette/nozzle neighbourhood, it is worth proceeding with exploring the placement sequence's neighbourhood since the placement sequence cost is dependent on the placement pipette/nozzle assignment. Finally, the sub tour sequence's neighbourhood is explored, as it is not really dependent on other neighbourhoods. The changes on the other neighbourhoods do not have a big impact to the sub tour sequence's neighbourhood. However, there is no clue to determine which neighbourhood's exploring sequence is the best since it is very problem specific. In fact, it might also rely on the solution space.

7.3.4 The Variable Neighbourhood Monte Carlo Search (VNMS)

To further explore the VNS approach, we have introduced another variant of VNS that is the Variable Neighbourhood Monte Carlo Search (VNMS), which differs from VNS-3. VNMS is also a descent-ascent heuristic that accepts an improved solution but probabilistically accepts a worse solution based on the EMCQ (Exponential Monte Carlo with counter, Q) acceptance criterion that operates at the higher level. VNMS has a three stages neighbourhood search, these being a steepest descent, EMC (Exponential Monte Carlo) and shaking. The idea is to initially explore the best neighbour in each neighbourhood structure before applying a random element. The search direction changes when the steepest descent local search that returns the best neighbour is unable to find an improved solution. At this stage, we apply an EMC local search for a number of iterations to obtain a good solution from a distant neighbourhood. The shaking procedure is only applied after the EMC local search cannot find an improved solution. In the shaking procedure, a neighbour solution is randomly generated from a set of neighbourhood structures until the new generated solution is accepted by the acceptance criterion of the VNMS (an EMCQ acceptance criterion). The VNMS algorithm is described in figure 7.4 and the flowchart is shown in figure 7.5. The novelty of the VNMS approach (in the context of VNS) is the concept of three stages of neighbourhood search, using an EMCQ acceptance criterion at the VNS level. The shaking procedure is only applied when the local searchers cannot find an improved solution. Let us denote x and f(x) as the incumbent solution and the quality of the incumbent solution, respectively.

```
Step A: (Initialisation)
        (1) Select the set of neighbourhood structures $n_d$, $d=1,2…,D$ , $n_e$,
            $e=1,2…,E$, $n_z$, $z=1,2…,Z$ that will be used in the search; find an
            initial solution x; choose a stopping condition;
        (2) Record the best obtained solution, $x_{best} \leftarrow x$ and $f(x_{best}) \leftarrow f(x)$;

Step B: (Steepest Descent Neighbourhood Search)
        (1) Set $d \leftarrow 1$;
        (2) Do
                a). Exploration of neighbourhood. Find the best neighbour, x' from
                    the $d^{th}$ neighbourhood of $x(x' Є n_d(x))$;
                b). Acceptance criteria (Move or not). Apply an EMCQ acceptance
                    criteria. If x' is accepted, then $x \leftarrow x'$.
                c). If the new solution is better than the incumbent solution,
                    continue the search with $n_d$; otherwise, set $d \leftarrow d+1$.
                d). If $f(x')<f(x_{best})$ then $x_{best} \leftarrow x'$ and $f(x_{best}) \leftarrow f(x')$;
            Until d>D or the stopping condition is met.

Step C: (EMC Neighbourhood search)
        (1) Set $e \leftarrow 1$;
        (2) Do
                a). Local search.Systematically generate a neighbour, x" from the
                    $e^{th}$ neighbourhood of $x(x" Є n_e(x))$ using EMC local search.
                    Return the best obtained solution x';
                b). Apply the same acceptance criteria as in Step B(2b);
                c). If the new solution is better than the incumbent, continue the
                    search with $n_e$; otherwise, set $e \leftarrow e+1$.
                d). If $f( x')<f(x_{best})$ then $x_{best} \leftarrow x'$ and $f(x_{best}) \leftarrow f(x')$;
            Until e>E or the stopping condition is met.

Step D: (Shaking)
        (1) Randomly select neighbourhood structures $n_z$,;
        (2) Do
                a). Generate a point x' at random from the $z^{th}$ neighbourhood of x
                    $(x' Є n_z(x))$;
                b). Apply the same acceptance criteria as in Step B(2b);
                c). If $f(x')<f(x_{best})$ then $x_{best} \leftarrow x'$ and $f(x_{best}) \leftarrow f(x')$. If x' is accepted,
                    then goto step B;
            Until x' is accepted or the stopping condition is met.

Step E: (Termination)
Goto Step B if stopping condition=false, otherwise terminate.
```

Figure 7.4: The proposed VNMS algorithm for component pick-and-place sequencing of multi-head SMD placement machine (minimisation problem).


Choose a starting solution.
1
Optimise the sub-tour's assignment using SD-LS.
Yes
δ<0
No
Optimise the component's picking using SD-LS.
Yes
δ<0
No
Optimise the component's placing using SD-LS.
Yes
δ<0
No
Optimise the sub-tours sequencing using SD-LS.
Yes
δ<0
No
Optimise the sub-tour's assignment using EMC-LS.
Yes
δ<0
No
Optimise the component's picking using EMC-LS.
δ<0
Yes
No
Optimise the component's placing using EMC-LS.
δ<0
Yes
No
Optimise the sub-tours sequencing using EMC-LS.
δ<0
Yes
No
Apply the shaking procedure using 3-opt LS.
Accept the solution
Yes
No
Stopping condition is true?
No
Yes
Return the best obtained solution

Note: δ=f(x')-f(x); SD-LS: Steepest descent local search; EMC-LS: EMC local search.

Figure 7.5: The flowchart of the variable neighbourhood search algorithm for optimising the component pick-and-place sequencing of multi-head SMD placement machine.

The steepest descent neighbourhood search has *D* (six in this work) steepest descent local searches. Each steepest descent local search will explore a different neighbourhood structure and returns the best neighbour. The returned solution will be accepted based on the EMCQ acceptance criterion at the VNMS level.

After exploring the steepest descent neighbourhood structures, VNMS will explore the EMC neighbourhood structures. The EMC neighbourhood search is designed to divert the search direction by probabilistically accepting some worse solution. This stage has *E* (six in this work) EMC local searches. These EMC local searches are the same as the EMC local searches in VNS-2 and VNS-3. Indeed, the VNS-3 and the EMC neighbourhood search stage are the same.

Finally, VNMS will apply the shaking stage if the steepest descent neighbourhood search and EMC neighbourhood search stages are unable to return an accepted solution. In this work, we use a 3-opt operator in the shaking procedure. As the problem domain involves six sub-problems, then we apply 3-opt operator for each of the sub-problems. Each trial solution obtained by the shaking procedure will be evaluated based on the VNMS acceptance criterion. The shaking procedure is repeated until the obtained solution is accepted or the stopping condition is met. Once the obtained solution is accepted, VNMS approach will continue the search by repeating the whole procedure again starting from Step B in figure 7.4, steepest descent neighbourhood search.

7.3.5 Local Search for Variable Neighbourhood Monte Carlo Search

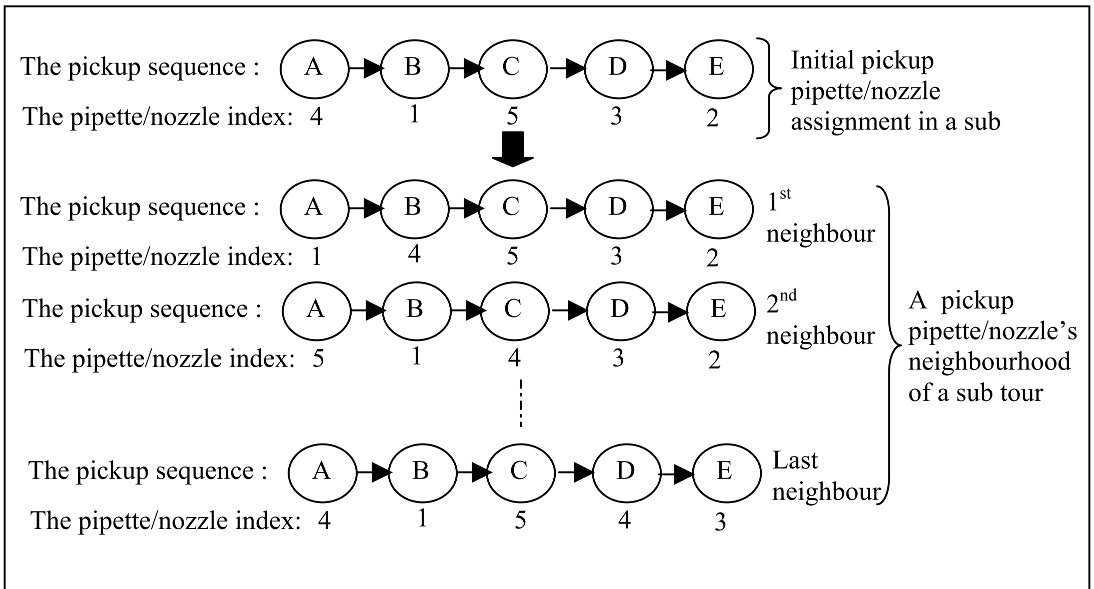
VNMS is a variable neighbourhood search with an EMCQ acceptance criterion. VNMS operates on eighteen neighbourhood structures, which are grouped into three sets of neighbourhoods. The three sets are based on the local searches that are steepest descent, EMC and random 3-opt operator. The difference is the way these neighbourhoods are explored and the neighbourhood size. The neighbourhoods are:

(1) Steepest descent neighbourhoods, n_d where*,* $d=1,2...,D$ (in this work D=6). This group consists of six neighbourhoods that are:

a) The sub tour's neighbourhood changes the assignment of PCB points to a sub tour by applying a 2-opt operator. All possible neighbours are explored. The neighbours in the neighbourhood are obtained by swapping each point in a sub tour with each point in the other sub tour by applying a 2–opt operator. A neighbour having the best solution's quality will be selected (returned to the VNMS).

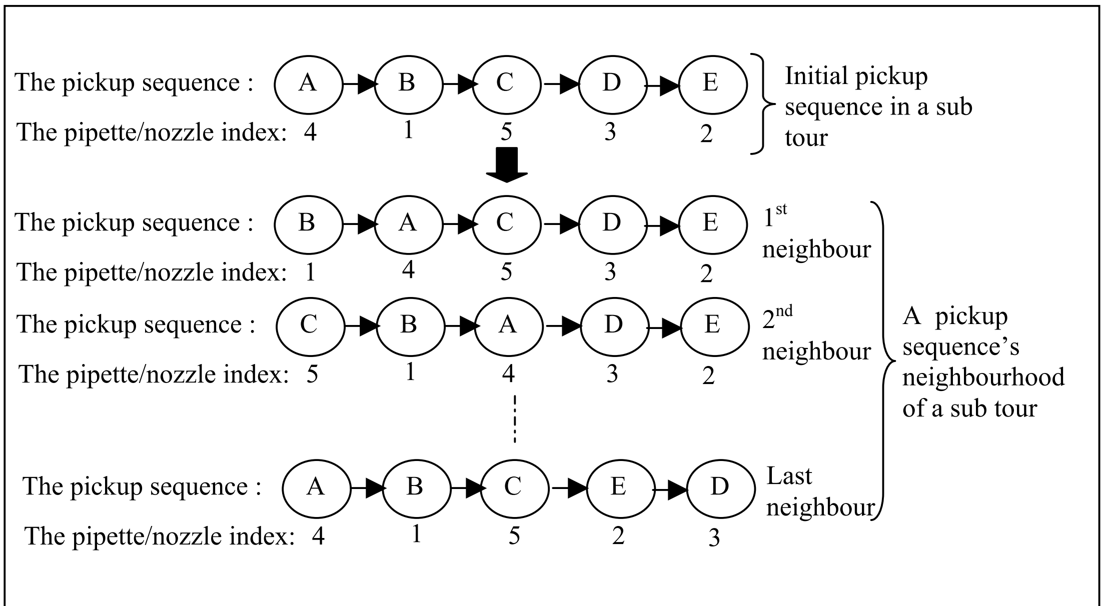
b) The pickup pipette/nozzle's neighbourhood changes the assignment of a pickup pipette/nozzle by applying a 2-opt operator. The neighbours are defined by swapping a pickup pipette/nozzle with each other pickup pipette/nozzle in the same sub tour by applying a 2-opt operator. If the swapping operations involve two different component types, then we modify the appropriate placement pipette/nozzle in the sub tour such that the PCB point will receive the correct component type. The neighbourhood move is shown in figure 7.6. The best neighbour in each sub tour will be selected. As the swapping operations are only performed within a sub tour, each sub tour has a unique neighbourhood. Therefore, accepting a solution in a sub tour does not change the neighbourhood in the other sub tours. Hence, we firstly accept the best neighbour in the first sub tour. Then, consecutively accept all the best neighbours in the following sub tours. As a result, the best obtained solution that will be returned to VNMS is generated by consecutive moves from the best neighbour of the first sub tour until the last sub tour. The first, second etc. sub tour are referring to the index of sub tour sequence.

c) The pickup sequence's neighbourhood changes the pickup sequence. The neighbours are defined by swapping a pickup sequence with each other pickup sequences in the same sub tour by applying a 2-opt operator. The pipette/nozzle assignment does not change. Figure 7.7 shows an example of the neighbourhood. Each sub tour has a unique neighbourhood. All neighbours for each sub tour neighbourhood will be visited. Similar to the pickup pipette/nozzle's neighbourhood, the best obtained solution that will be returned to the VNMS is generated by consecutive moves from the best neighbour of the first sub tour until the last sub tour.


The pickup sequence : A B C D E
The pipette/nozzle index: 4 1 5 3 2
Initial pickup pipette/nozzle assignment in a sub
The pickup sequence : A B C D E
The pipette/nozzle index: 1 4 5 3 2
1st neighbour
The pickup sequence : A B C D E
The pipette/nozzle index: 5 1 4 3 2
2nd neighbour
A pickup pipette/nozzle's neighbourhood of a sub tour
The pickup sequence : A B C D E
The pipette/nozzle index: 4 1 5 4 3
Last neighbour

Note: A, B, C, D, E represent the nodes (i.e. pickup components) and the arrows represent the pickup's sequence.

Figure 7.6: An example of a pickup pipette/nozzle's neighbourhood of a sub tour by applying 2-opt operator.


The pickup sequence : A B C D E
The pipette/nozzle index: 4 1 5 3 2
Initial pickup sequence in a sub tour
The pickup sequence : B A C D E
The pipette/nozzle index: 1 4 5 3 2
1st neighbour
The pickup sequence : C B A D E
The pipette/nozzle index: 5 1 4 3 2
2nd neighbour
A pickup sequence's neighbourhood of a sub tour
The pickup sequence : A B C E D
The pipette/nozzle index: 4 1 5 2 3
Last neighbour

Note: A, B, C, D, E represent the nodes (i.e. pickup components) and the arrows represent the pickup's sequence.

Figure 7.7: An example of a pickup sequence's neighbourhood of a sub tour by applying 2-opt operator.

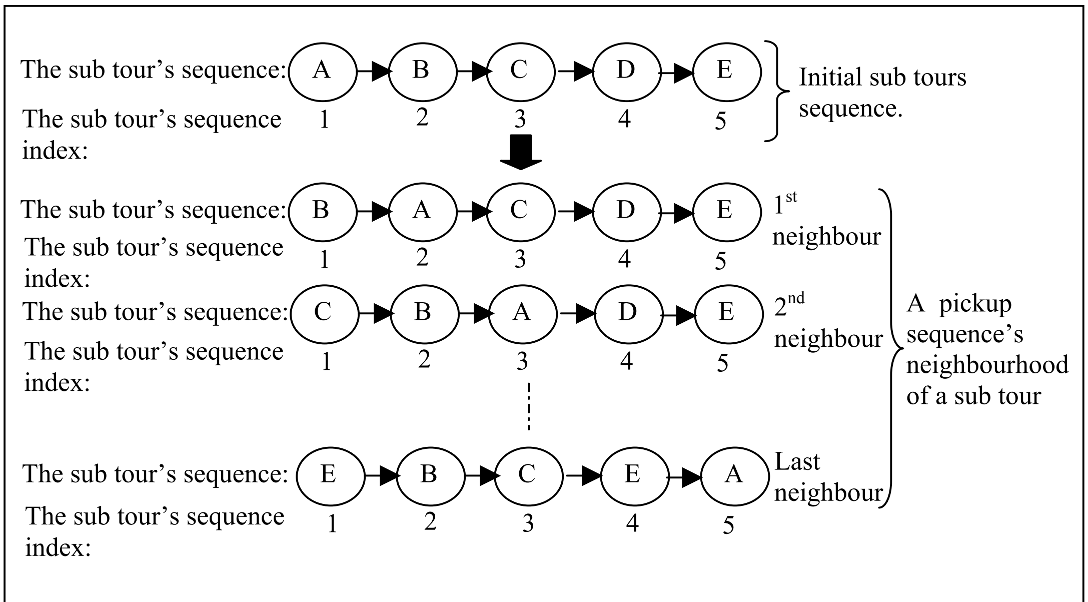
d) The placement pipette/nozzle's neighbourhood changes the assignment of the placement pipette/nozzle by applying a 2-opt operator. If the swapping operations involve two different component types, then we modify the appropriate pickup pipette/nozzle in the sub tour such that the component will be picked up with the correct pipette/nozzle. The neighbours in this neighbourhood are obtained in a similar way of obtaining the pickup pipette/nozzle's neighbourhood except that instead of swapping pickup's pipette/nozzle, we swap the placement's pipette/nozzle. The best obtained solution that will be returned to VNMS is also generated by consecutive moves from the best neighbour of the first sub tour until the last sub tour.

e) The placement sequence's neighbourhood changes the placement sequence by applying a 2-opt operator. The neighbours in this neighbourhood are obtained in a similar way of obtaining the pickup sequence's neighbourhood except that instead of swapping pickup's sequence, we swap the placement's sequence. The best obtained solution that will be returned to VNMS is also generated by consecutive moves from the best neighbour of the first sub tour until the last sub tour.

f) The sub tour sequence's neighbourhood changes the sequence order of the sub tours by applying a 2-opt operator. The neighbour is defined by exchanging the index sequence of two sub tours. This is shown in figure 7.8. All the neighbours in the neighbourhood will be visited. The best neighbour solution will be returned to VNMS.

(2) EMC neighbourhoods, n_e where, $e=1,2...,E$ (in this work E=6). These neighbourhoods are almost identical to the neighbourhoods used in the VNS-2 and VNS-3 except that instead of randomly choosing a sub tour to be explored, this neighbourhood includes all the available sub tours. There are also six neighbourhoods as in VNS-2 and VNS-3. The best solution obtained from the visited sub tours will be used as an incumbent solution for the subsequent sub tour. The best obtained solution that will be returned to

VNMS is also generated by consecutive moves from the best obtained solution of the first sub tour until the last sub tour.



The sub tour's sequence: A B C D E
The sub tour's sequence index: 1 2 3 4 5
Initial sub tours sequence.
The sub tour's sequence: B A C D E
The sub tour's sequence index: 1 2 3 4 5
1st neighbour
The sub tour's sequence: C B A D E
The sub tour's sequence index: 1 2 3 4 5
2nd neighbour
A pickup sequence's neighbourhood of a sub tour
The sub tour's sequence: E B C E A
The sub tour's sequence index: 1 2 3 4 5
Last neighbour

Note: A, B, C, D, E represent the nodes (i.e. sub tours) and the arrows represent the sub tours sequence.

Figure 7.8: An example of a sub tour sequence's neighbourhood by applying 2-opt operator.

(3) Random 3-opt neighbourhoods, n_z where, $z=1,2...,Z$ (in this work Z=6). There are also six neighbourhoods, these being the sub tour's, pickup pipette/nozzle's, pickup sequence's, placement pipette/nozzle's, placement sequence's and sub tour sequence's neighbourhoods. The neighbour is generated by applying a 3-opt operator. Only one neighbour solution will be visited for every calling sequence. The three points that are randomly chosen to be swapped, are selected from the same sub tour (for generating a neighbour of pickup pipette/nozzle, pickup sequence, placement pipette/nozzle or placement sequence) or a different sub tour (for generating a neighbour of sub tour or sub tour's sequence). Figure 7.9 shows an example of generating a neighbour in a pickup pipette/nozzle's neighbourhood.

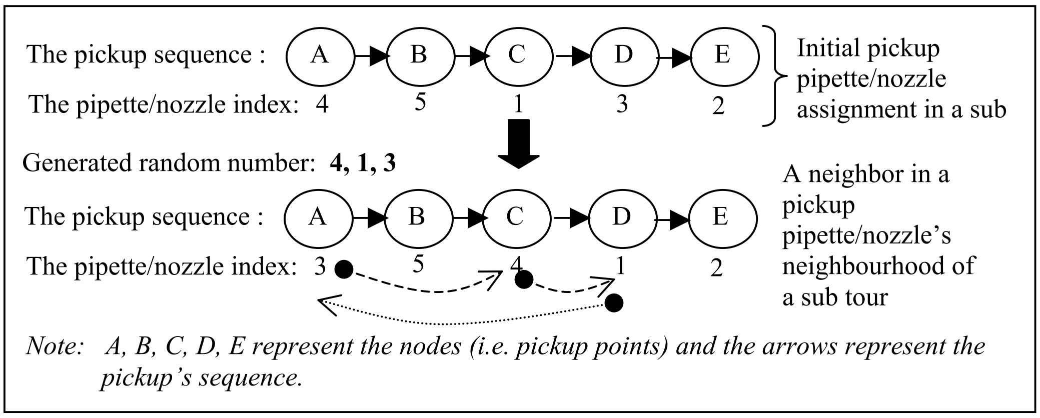

The pickup sequence :
A
B
C
D
E
The pipette/nozzle index: 4 5 1 3 2
Initial pickup pipette/nozzle assignment in a sub
Generated random number: 4, 1, 3
The pickup sequence :
A
B
C
D
E
The pipette/nozzle index: 3 5 4 1 2
A neighbor in a pickup pipette/nozzle's neighbourhood of a sub tour
Note: A, B, C, D, E represent the nodes (i.e. pickup points) and the arrows represent the pickup's sequence.

Figure 7.9: An example of generating a pickup pipette/nozzle's neighbour by using a 3-opt operator.

The local searchers used in the VNMS are listed in table 7.1. H_c denotes the heuristic ID whilst the acceptance criteria determine how to accept the solution in the local search. As the problem domain requires six different swapping operations, we developed three sets of neighbourhood structures, having six swapping operations in each set to produce six different neighbourhood structures in each set. The three sets are based on the local searches that are steepest descent (SD-LS), Exponential Monte Carlo (EMC-LS) and random 3-opt operator. The steepest descent neighbourhood search stage uses six local searchers, these being H_c={0,1,2,3,4,5}. The EMC neighbourhood search stage uses the next six local searchers that are H_c={6,7,8,9,10,11} whereas the other six local searchers (i.e. H_c={12,13,14,15,16,17}) that are actually not a local search but just a simple 3-opt operators, are used in the shaking procedure.

7.4 Hyper-heuristic

As this is an unexplored problem, it is difficult to evaluate the performance of the VNMS approach against data from the literature. Therefore, in this work, we have developed nine hyper-heuristic approaches for comparison purposes. These hyper-heuristics operate on different sets of low-level heuristics (LLHs or local searchers) and acceptance criteria:

a) AM-sdEmc3opt (All Move that operates with SD, EMC and 3-opt LLH): Randomly select LLH and accept any solution returned by the LLH. There are eighteen LLH (these are the same local searchers used in VNMS approach).

TABLE 7.1: A LIST OF LOCAL SEARCHES USED IN VARIABLE NEIGHBOURHOOD MONTE CARLO SEARCH (VNMS)

H_c	**Local search name**	**Acceptance Criteria**
0	Swap placement sub-tour using SD-LS	choose the best
1	Swap pickup pipette/nozzle using SD-LS	choose the best
2	Swap pickup sequence using SD-LS	choose the best
3	Swap placement pipette/nozzle using SD-LS	choose the best
4	Swap placement sequence using SD-LS	choose the best
5	Swap sub-tour's sequencing order using SD-LS	choose the best
6	Swap placement sub-tour using EMC-LS	$e^{-\theta/\tau}$ where $\theta=\delta*t$ and $\tau=Q$
7	Swap pickup pipette/nozzle using EMC-LS	$e^{-\theta/\tau}$ where $\theta=\delta*t$ and $\tau=Q$
8	Swap pickup sequence using EMC-LS	$e^{-\theta/\tau}$ where $\theta=\delta*t$ and $\tau=Q$
9	Swap placement pipette/nozzle using EMC-LS	$e^{-\theta/\tau}$ where $\theta=\delta*t$ and $\tau=Q$
10	Swap placement sequence using EMC-LS	$e^{-\theta/\tau}$ where $\theta=\delta*t$ and $\tau=Q$
11	Swap sub-tour's sequencing order using EMC-LS	$e^{-\theta/\tau}$ where $\theta=\delta*t$ and $\tau=Q$
12	Swap placement sub-tour using random 3-opt	none
13	Swap pickup pipette/nozzle using random 3-opt	none
14	Swap pickup sequence using random 3-opt	none
15	Swap placement pipette/nozzle using random 3-opt	none
16	Swap placement sequence using random 3-opt	none
17	Swap sub-tour's sequencing order using random 3-opt	none

b) OI-sdEmc3opt (Only Improving that operates with SD, EMC and 3-opt LLH): Randomly select LLH and only accept an improved solution returned by the LLH.

c) OICF-sdEmc3opt (Only Improving Choice Function that operates with SD, EMC and 3-opt LLH): Select LLH based on historical performance (Cowling et al., 2001a, b) and only accepts an improved solution returned by the LLH.

d) AMCF-sdEmc3opt (All Move Choice Function that operates with SD, EMC and 3-opt LLH): Same as OICF-sdEmc3opt but in this case we accept all solutions returned by the LLH's.

e) EMCQ-sdEmc3opt (Exponential Monte Carlo with Counter that operates with SD, EMC and 3-opt LLH): Randomly select LLH and accepts the returned solution based on the EMCQ acceptance criteria.

f) AM-sd (All Move that operates with SD LLH): Randomly select LLH and accept any solution returned by the LLH. There are only six steepest descent LLH (that are the same steepest descent local searches used in VNMS).

g) OI-2opt (Only Improving that operates with 2-opt LLH): Randomly select LLH (i.e. simple 2-opt operator) and only accept an improved solution returned by the LLH. There are six simple 2-opt operators as used in chapter 5.

h) OICF-2opt (Only Improving Choice Function that operates with 2-opt LLH): Select LLH based on historical performance (Cowling et al., 2001a, b) and only accepts an improved solution returned by the LLH.

i) EMCQ-2opt (EMCQ that operates with 2-opt LLH): Randomly select LLH and accepts the returned solution based on the EMCQ acceptance criteria.

Table 7.2 shows the difference among these hyper-heuristics and the variable neighborhood search approaches.

7.5 Testing and Results

An initial solution is generated using either a randomised or an ordered constructive heuristic that we proposed in section 5.3.1. We use the same assumptions as in chapter 5. Since we model the same placement machine as in chapter 5, we also apply the same experimental parameters. In the experiment we modelled the theoretical multi-head SMD placement machine that has a head equipped with 8 pipette/nozzles.

TABLE 7.2: A LIST OF HYPER-HEURISTICS AND VNS APPROACHES WITH THEIR LOCAL SEARCH (OR LLH), ACCEPTANCE CRITERIA AND CALLING SEQUENCE.

Heuristic	**Local search (or LLH)**	**Acceptance criteria**	**Calling sequence**
VNS-1	RD-LS	Descent	Systematic
VNS-2	EMC-LS	Descent	Systematic
VNS-3	EMC-LS	EMCQ	Systematic
VNMS	SD-LS+EMC-LS+3opt	EMCQ	Systematic
AM-sdEmc3opt	SD-LS+EMC-LS+3opt	Accept all moves	Random
OI-sdEmc3opt	SD-LS+EMC-LS+3opt	Descent	Random
OICF-sdEmc3opt	SD-LS+EMC-LS+3opt	Descent	Based on historical performance
AMCF-sdEmc3opt	SD-LS+EMC-LS+3opt	Accept all moves	Based on historical performance
EMCQ-sdEmc3opt	SD-LS+EMC-LS+3opt	EMCQ	Random
AM-sd	SD-LS	Accept all moves	Random
OI-2opt	2-opt	Descent	Random
OICF-2opt	2-opt	Descent	Based on historical performance
EMCQ-2opt	2-opt	EMCQ	Random

In the experiment we use four datasets (dataset N80K20_A, N80K20_D, N240K40_D and N240K40_F). The specification of these datasets is shown in table 7.3. These datasets (see Appendix A) are randomly generated using our random PCB generator software called PCBgen.

TABLE 7.3: EXPERIMENTAL DATASETS

Dataset	**N**	**K**	**BL**	**BW**
N80K20_A	80	20	600	200
N80K20_D	80	20	2000	1000
N240K40_D	240	40	2000	1000
N240K40_F	240	40	1800	600

We ran the experiments using the same PC as in chapter 6. As in chapter 6, we set *α=1.0, β=0.01 and σ=0.5 (for OICF and AMCF)*. Other approaches are not parameter sensitive. Table 7.4a and 7.4b show the experimental results over an average of thirty runs on datasets N80K20_A, N80K20_D and N240K40_D, N240K40_F, with each run being given ten minutes of computation time as a termination criterion. However, any other termination criterion would also be suitable. For example, if we apply these methods for an on-line scheduling problem that we proposed in chapter 5, we might be able to continually search for an improved schedule until there are no more PCB's to be assembled. The initial cycle time (CT) for dataset N80K20_A, N80K20_D, N240K40_D and N240K40_F are 1061.04, 3238.90, 9693.76 and 8385.98 unit time, respectively. The figures in table 7.4 show the average of the best obtained solution's qualities (CT_{avg}), the percentage of the CT's improvement over the initial CT (I%), standard deviation of CT (Dev) and the minimum CT obtained over 30 runs (CT_{min}).

TABLE 7.4A: AN AVERAGE RESULT OF THIRTY RUNS ON EACH DATASET (TEST DURATION: 10 MINUTES)

Heuristic	**Dataset N80K20_A (CT$_o$=1061.04)**				**Dataset N80K20_D (CT$_o$=3238.90)**			
	CT$_{avg}$	**I(%)**	**Dev**	**CT$_{min}$**	**CT$_{avg}$**	**I (%)**	**Dev**	**CT$_{min}$**
VNMS	**267.38**	**74.80**	**3.00**	**261.68**	**781.41**	**75.87**	**4.58**	**769.56**
VNS-1	313.67	70.44	13.98	286.73	840.32	74.06	29.28	772.73
VNS-2	307.22	71.05	12.68	282.26	829.54	74.39	30.91	769.93
VNS-3	295.03	72.19	11.68	276.98	810.58	74.97	25.34	760.70
AM-sdEmc3opt	280.70	73.55	7.27	266.38	763.34	76.43	17.50	734.04
OI-sdEmc3opt	282.40	73.38	11.39	262.24	786.18	75.73	28.73	754.63
AMCF-sdEmc3opt	274.04	74.17	10.93	258.30	755.44	76.68	30.18	712.95
OICF-sdEmc3opt	282.21	73.40	11.52	261.99	776.38	76.03	21.48	732.15
EMCQ-sdEmc3opt	281.04	73.51	9.92	257.64	784.41	75.78	26.97	742.93
AM-sd	284.45	73.19	10.87	260.89	813.48	74.88	40.93	763.79
EMCQ-2opt	282.91	73.34	7.30	266.74	772.01	76.16	24.95	733.55
OICF-2opt	343.18	67.66	32.38	298.46	931.62	71.24	147.61	772.58
OI-2opt	308.00	70.97	12.92	281.13	840.41	74.05	45.37	781.75

TABLE 7.4B: AN AVERAGE RESULT OF THIRTY RUNS ON EACH DATASET (TEST DURATION: 10 MINUTES)

Heuristic	**Dataset N240K40_D (CT_o=9693.76)**				**Dataset N240K40_F (CT_o=8385.98)**			
	CT_{avg}	**I(%)**	**Dev**	**CT_{min}**	**CT_{avg}**	**I(%)**	**Dev**	**CT_{min}**
VNMS	**2637.02**	**72.80**	**0.00**	**2637.02**	**2274.50**	**72.88**	**0.00**	**2274.50**
VNS-1	3629.63	62.56	142.75	3219.32	3129.98	62.68	125.44	2949.36
VNS-2	3589.54	62.97	156.19	3266.36	3112.83	62.88	108.71	2928.57
VNS-3	3597.52	62.89	128.34	3322.07	3102.69	63.00	98.76	2940.86
AM-sdEmc3opt	2683.72	72.31	75.71	2555.92	2327.52	72.25	58.40	2257.42
OI-sdEmc3opt	2670.66	72.45	89.90	2500.86	2300.66	72.57	64.75	2175.72
AMCF-sdEmc3opt	2645.29	72.71	63.88	2525.14	2271.15	72.92	52.96	2170.43
OICF-sdEmc3opt	2623.91	72.93	53.83	2523.32	2269.05	72.94	55.96	2150.11
EMCQ-sdEmc3opt	2672.56	72.43	77.22	2522.60	2332.49	72.19	60.68	2218.04
AM-sd	2664.36	72.51	75.10	2476.90	2309.31	72.46	52.97	2226.15
EMCQ-2opt	3222.23	66.76	97.10	3039.54	2782.71	66.82	89.65	2614.95
OICF-2opt	3698.55	61.85	657.87	2742.89	3318.26	60.43	619.44	2402.04
OI-2opt	3352.15	65.42	115.04	3121.82	2989.95	64.35	156.19	2769.64

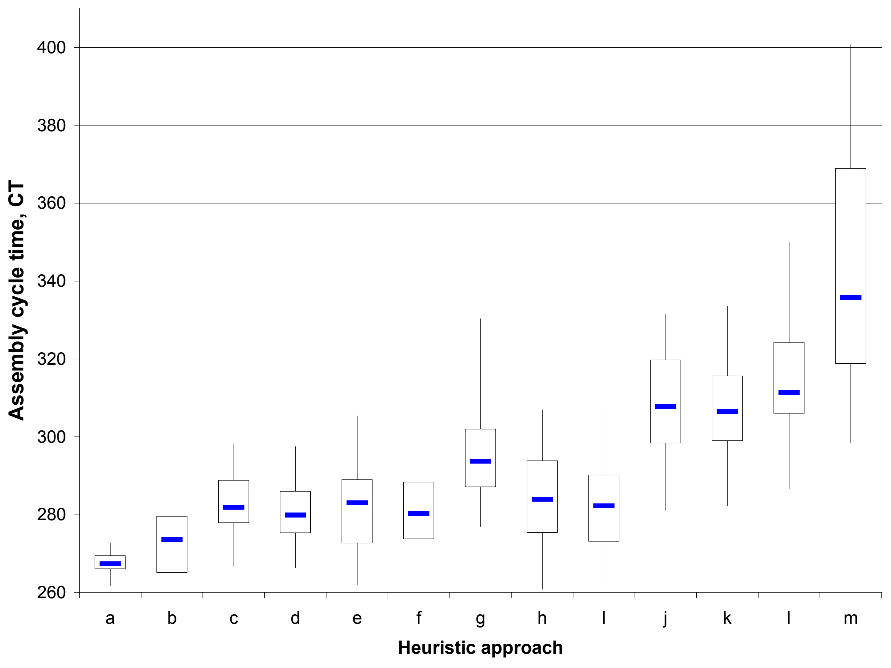
Note: *CT_{avg}=Average assembly cycle time(unit time); CT_{min}=Minimum assembly cycle time(unit time);*
Dev=standard deviation of CT; CT_o=Initial CT;
*I=CT's Improvement=(CT_o-CT_{avg})*100/ CT_o*

The results in table 7.4, table 7.5 and figure 7.10, show that the VNMS and the hyper-heuristics that operate on SD-LS, EMC-LS and 3-opt LLH generally show fairly equal performance in respect to the CT's average over thirty runs (10 minutes test duration).

TABLE 7.5: RESULT OF THIRTY RUNS ON DATASET N80K20_A (TEST DURATION: 10 MINUTES)

Heuristic	**CT$_{min}$**	**CT$_{avg}$**	**CT$_{max}$**	**Dev**	**Mode**	**Median**	**Skewness**
VNMS	**261.68**	**267.38**	272.73	**3.00**	267.00	267.00	-0.55
VNS-1	286.73	313.67	350.03	13.98	325.00	311.00	0.53
VNS-2	282.26	307.22	333.65	12.68	299.00	307.00	0.18
VNS-3	276.98	295.03	330.35	11.68	287.00	294.00	0.89
AM-sdEmc3opt	266.38	280.70	297.51	7.27	279.00	280.00	0.17
OI-sdEmc3opt	262.24	282.40	308.51	11.39	271.00	282.00	0.57
AMCF-sdEmc3opt	258.30	274.04	305.76	10.93	265.00	273.00	1.00
OICF-sdEmc3opt	261.99	282.21	305.39	11.52	285.00	283.00	0.30
EMCQ-sdEmc3opt	257.64	281.04	304.73	9.92	279.00	280.00	0.09
AM-sd	260.89	284.45	306.95	10.87	275.00	284.00	0.18
EMCQ-2opt	266.74	282.91	298.21	7.30	275.00	282.00	0.34
OICF-2opt	298.46	343.18	400.68	32.38	337.00	336.00	0.47
OI-2opt	281.13	308.00	331.46	12.92	295.00	308.00	-0.09

This is the expected result since they are dealing with the same local searches. In fact, the ANOVA (analysis of variance) on dataset N80K20_A (10 minutes test duration) demonstrated that the difference in CT's averages of all approaches tested in this section are statistically not significant (with 99% confident level). This result supports our argument in chapter 6 in which we said, "simple hyper-heuristics (such as AM) and more complex hyper-heuristics (such as EMC and EMCQ) might produce fairly similar results when using a *complex* set of LLH's". This is due to the fact that the *complex* set of LLH might be able to find good solutions without having to be guided by a hyper-heuristic whilst a set of *simple* LLHs might provide more flexibility for the hyper-heuristic. The AM-sd hyper-heuristic also shows an almost equal performance as VNMS. This indicates that the search direction might be dictated by the SD local searches. In fact, the SD local search is computationally expensive since it has to visit all the neighbours in each neighbourhood in order to find the best neighbour. Perhaps, the SD local searches might indirectly dominate most of the searching time especially early in the search, where there is possibly a lot of room for improvement.



Note:	a= VNMS;	b=AMCF-sdEmc3opt;	c=EMCQ-2opt
	d=AM-sdEmc3opt;	e=OICF-sdEmc3opt;	f=EMCQ-sdEmc3opt;
	g=VNS-3;	h=AM-sd;	i=OI-sdEmc3opt;
	j=OI-2opt;	k=VNS-2;	l=VNS-1;
	m=OICF-2opt.		

Figure 7.10: A comparison of VNMS, VNS and hyper-heuristic approaches using box-and-whisker plot on the results of ten runs on dataset N80K20_A (test duration: 10 minutes).

Therefore, we further investigated the performance of these approaches for longer runtimes. Table 7.6 shows the experimental results of the average of ten runs on datasets N80K20_A and N240K40_F with each run being given one hour of computation time as a termination criterion.

In contrast to the case above, an ANOVA test on the results of dataset N80K20_A and N240K40_F for longer runtimes (one hour duration) showed that the CT's averages of all approaches tested in this section are statistically different (at the 99% confidence level), (F-ratio=47.87 and 33.27, respectively for α=0.01). Therefore, by just looking at the CT's averages, we can briefly estimate the effectiveness of the approaches (shown in table 7.7).

Based on the results in table 7.7, we can observe that VNMS and AMCF-sdEmc3opt show the best performance on dataset N80K20_A and N240K40_F, respectively). Interestingly, VNMS and AMCF-sdEmc3opt have obtained the best performance of the two heuristics ranked in this experiment (on dataset N80K20_A and N240K40_F). However, based on an ANOVA test carried out on the results of EMCQ-2opt, AMCF-sdEmc3opt and VNMS there is no significant difference in their performance (F-ratio=0.39 for α=0.01) even though they used different local searchers. To further analyse the effect of the local searchers (i.e. the low-level heuristics for the hyper-heuristic approaches) applied in each approach, we performed several other ANOVA tests (on the results of dataset N80K20_A for one hour runtimes) as follows (all the ANOVA tests are based on the average cycle times and standard deviations of the cycle times):

a) For the ANOVA test on the results of VNMS, AMCF-sdEmc3opt, AM-sdEmc3opt, OICF-sdEmc3opt, EMCQ-sdEmc3opt and OI-sdEmc3opt approaches there is a significant difference (F-ratio=17.23 for α=0.01). This indicates that there is an effect of using different higher-level heuristics on the same local search. Therefore, by just referring to the average cycle times, we can observe that VNMS shows the best performance over the other approaches (AMCF-sdEmc3opt, AM-sdEmc3opt, OICF-sdEmc3opt, EMCQ-sdEmc3opt and OI-sdEmc3opt).

b) For the ANOVA test on the results of EMCQ-2opt, OI-2opt and OICF-2opt hyper-heuristic approaches there is a significant difference (F-ratio=78.13 for α=0.01). Again, this indicates that there is an effect of using different higher-level heuristics on the same local search. In this group, the EMCQ-2opt outperformed the other methods (i.e. OI-2opt and OICF-2opt).

c) For the ANOVA test on the results of VNS-3 and VNS-2 the result is significantly different (F-ratio=32.74 for α=0.01). This indicates that VNS-3 is superior to VNS-2.

d) For the ANOVA test on the results of VNS-1 and VNS-2 the result is significantly different (F-ratio=8.56 for α=0.01). This indicates that there is an effect of applying different local searchers on the same higher-level heuristic. In this case VNS-2 is superior to VNS-1.

e) For the ANOVA test on the results of EMCQ-2opt and EMCQ-sdEmc3opt approaches the result is significantly different (F-ratio=21.48 for α=0.01). Again, this indicates the there is an effect of applying different local searchers on the same higher-level heuristic. In this group, EMCQ-2opt is superior to EMCQ-sdEmc3opt.

TABLE 7.6. AN AVERAGE RESULT OF TEN RUNS ON EACH DATASET (TEST DURATION: 1 HOUR)

Heuristic	**Dataset N80K20_A**				**Dataset N240K40_F**			
	CT_{avg}	**I(%)**	**Dev**	**CT_{min}**	**CT_{avg}**	**I(%)**	**Dev**	**CT_{min}**
VNMS	**257.04**	**75.77**	**2.49**	**252.86**	**2052.12**	**75.53**	**0.00**	**2052.12**
VNS-1	311.61	70.63	9.63	300.10	2590.57	69.11	82.64	2477.19
VNS-2	298.65	71.85	10.18	286.40	2617.47	68.79	75.06	2482.07
VNS-3	275.74	74.01	7.53	263.19	2592.18	69.09	63.18	2480.02
AM-sdEmc3opt	272.58	74.31	4.78	264.10	2086.48	75.12	33.16	2045.16
OI-sdEmc3opt	278.26	73.77	9.06	262.15	2063.53	75.39	29.93	2031.71
AMCF-sdEmc3opt	258.63	75.62	6.28	251.85	2047.13	75.59	37.48	1970.74
OICF-sdEmc3opt	273.97	74.18	6.36	264.96	2070.48	75.00	37.84	2026.71
EMCQ-sdEmc3opt	274.47	74.13	9.48	259.00	2063.70	75.39	26.02	2007.75
AM-sd	276.98	73.90	12.19	256.68	2053.88	75.51	22.63	2008.94
EMCQ-2opt	258.77	75.61	4.98	252.14	2264.44	73.00	50.43	2213.25
OICF-2opt	325.42	69.33	18.92	296.95	2781.95	66.83	512.33	2151.95
OI-2opt	279.03	73.70	8.09	264.50	2369.73	71.74	75.24	2276.86

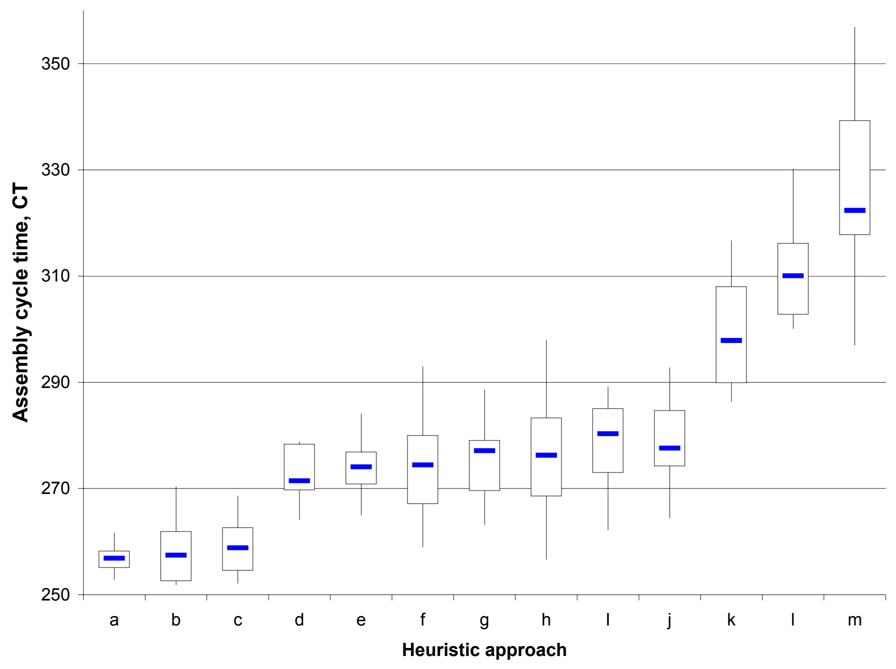
TABLE 7.7: A HEURISTIC RANKING BASED ON CT_{avg} OF TEN RUNS STARTING WITH THE MOST EFFECTIVE HEURISTIC (SMALLEST CT_{avg}) FOR ONE HOUR TEST DURATION

	Dataset N80K20_A		**Dataset N240K40_F**	
	Heuristic rank	**CT_{avg}**	**Heuristic rank**	**CT_{avg}**
1	VNMS	257.04	AMCF-sdEmc3opt	2047.13
2	AMCF-sdEmc3opt	258.63	VNMS	2052.12
3	EMCQ-2opt	258.77	AM-sd	2053.88
4	AM-sdEmc3opt	272.58	OI-sdEmc3opt	2063.53
5	OICF-sdEmc3opt	273.97	EMCQ-sdEmc3opt	2063.70
6	EMCQ-sdEmc3opt	274.47	OICF-sdEmc3opt	2070.48
7	VNS-3	275.74	AM-sdEmc3opt	2086.48
8	AM-sd	276.98	EMCQ-2opt	2264.44
9	OI-sdEmc3opt	278.26	OI-2opt	2369.73
10	OI-2opt	279.03	VNS-1	2590.57
11	VNS-2	298.65	VNS-3	2592.18
12	VNS-1	311.61	VNS-2	2617.47
13	OICF-2opt	325.42	OICF-2opt	2781.95

For this dataset (N80K20_A), we can conclude that the performance of the heuristic is dependent on both the local search and the higher-level heuristic. Apparently, the AMCF-sdEmc3opt shows better performance compared to OICF-sdEmc3opt whilst OICF-sdEmc3opt is better than OICF-2opt (which is the worst among all the approaches). This might indicate that the choice-function hyper-heuristic can work well with a set of complex and simple low-level heuristics (LLHs). The AMCF-sdEmc3opt, which accepts any solution returned from the LLHs, might provide more flexibility for the hyper-heuristic in choosing good LLHs (based on historical performance) and might diversify the search by accepting worse solutions, whilst the OICF-sdEmc3opt, with descent acceptance criterion, might restrict the search. Perhaps, the OICF-2opt, does not performed well due to the set of 2-opt LLHs, which have almost equal/stochastic performance. On the contrary, the set of 2-opt LLHs operating with the EMCQ hyper-heuristic (i.e. EMCQ-2opt) shows good performance. Again, this might be due to the fact that a set of *simple* LLHs might provide more flexibility for the hyper-heuristic.

For dataset N240K40_F (one hour runtime), the ANOVA test on the results of the AMCF-sdEmc3opt, VNMS, OICF-sdEmc3opt, EMCQ-sdEmc3opt, AM-sdEmc3opt, OI-sdEmc3opt and AM-sd approaches shows that these approaches have fairly equal performance (F-ratio=2.05 for α=0.01). Indeed, the CT's average for VNMS and AM-sd are almost equal and the standard deviation for VNMS is zero, which might indicate that the solution space for this dataset is very large and therefore the searching time is dominated by the SD-LS only without having a chance for the other local searchers to be called by the VNMS. Therefore, each run will produce the same solution because the SD-LS (that are systematically called by VNMS) will only return the best neighbour to the VNMS. Therefore, longer runtimes might be necessary to observe the performance of these approaches on this dataset.

Figure 7.11 shows a box-and-whisker plot, which represents the results of ten runs (minimum, first quartile, median, third quartile and maximum values of CT) on dataset N80K20_A (one hour runtime).By referring to figure 7.11, we can observe that the VNMS outperformed the other approaches with the smallest median and variation of the CT values. Figure 7.11 also shows that the AMCF-sdEmc3opt and EMCQ-2opt have fairly equal performance; and the AM-sdEmc3opt, OICF-sdEmc3opt, EMCQ-sdEmc3opt, VNS-3, Am-sd, OI-sdEmc3opt and OI-2opt also have fairly equal performance. These results also show that our basic VNS approaches are not performing well compared to the other approaches across all the datasets tested in this work. This may due to the local searches used in the basic VNS and the basic VNS approaches itself. Generally, the quality of the obtained solution by basic VNS is dependent on the local search used (Hansen and Mladenović, 1997). Unfortunately, the nature of our problem presents difficulties when applying other established local search such as tabu search and simulated annealing since this scheduling problem involves some interrelated sub problems, which should not be solved independently. There is also the issue of how far to solve each of the sub problems before solving the others and in which order should we solve the sub problem. Optimising one factor (sub problem) may increase the cost of another factor(s). However, we can conclude that the VNS may perform better by integrating an acceptance criterion at the VNS level. In addition, the strategy to initially explore the best neighbour in each neighbourhood structure before applying a random element and shaking procedures (which will only be applied after the first local search group cannot find an improved solution) might be effective. The drawback is that the steepest descent local search is very time consuming.



Note: *a= VNMS;* *b=AMCF-sdEmc3opt;* *c=EMCQ-2opt*
d=AM-sdEmc3opt; *e=OICF-sdEmc3opt;* *f=EMCQ-sdEmc3opt;*
g=VNS-3; *h=AM-sd;* *i=OI-sdEmc3opt;*
j=OI-2opt; *k=VNS-2;* *l=VNS-1;*
m=OICF-2opt.

Figure 7.11: A comparison of VNMS, VNS and hyper-heuristic approaches using box-and-whisker plot on the results of ten runs on dataset N80K20_A (test duration: 1 hour).

The behavior of the VNMS and hyper-heuristics over time can be observed in figure 7.12. The graph in figure 7.12 is plotted by randomly choosing one sample run on dataset N80K20_A for VNMS, AM-sdEmc3opt, AMCF-sdEmc3opt, OICF-sdEmc3opt, EMCQ-2opt, OICF-2opt and OI-2opt approaches. The points in the graph are plotted every 10 seconds for 10 minutes runs. The graph shows how these methods explore the search space and escape from local optima. It can be seen from figure 7.12 that the 2-opt hyper-heuristics (EMCQ-2opt, OICF-2opt and OI-2opt) are slightly better compared to the others in shorter timescale (less than 15 seconds). On the other hand, in longer time scales (greater than 15 seconds), the other methods such as AM-sdEmc3opt, AMCF-sdEmc3opt and OICF-sdEmc3opt, especially VNMS are outperformed compared to the 2-opt hyper-heuristics. However, the EMCQ-2opt, for example, is sometimes capable of finding a good solution for longer time scales.

Based on the result of this work, we can conclude that the VNMS is capable of producing good quality and stable results (for smaller dataset) in solving the component pick-and-place sequence of multi-head SMD placement machine. This means that the VNMS is more reliable compared to the hyper-heuristics approaches tested in this work.

7.6 Summary

A Variable Neighbourhood Search for solving the component placement sequencing of a theoretical multi-head SMD placement machine has been proposed in this chapter. A basic version of a VNS, that is VNS-1, VNS-2 and VNS-3 have been presented. The VNS-1 is a descent heuristic that operates with random descent local search. VNS-2 is also a descent heuristic but operates with EMC (Exponential Monte Carlo) local search. The EMC local search always accepts an improved solution and probabilistically accepts a worse solution depending on the degree of worsening (δ). The VNS-3 is a descent-ascent heuristic with EMCQ (Exponential Monte Carlo with counter) that operates with an EMC local search. Our experimental results show that the VNS-1, VNS-2 and VNS-3 do not perform well compared to some hyper-heuristics approaches presented in this work.

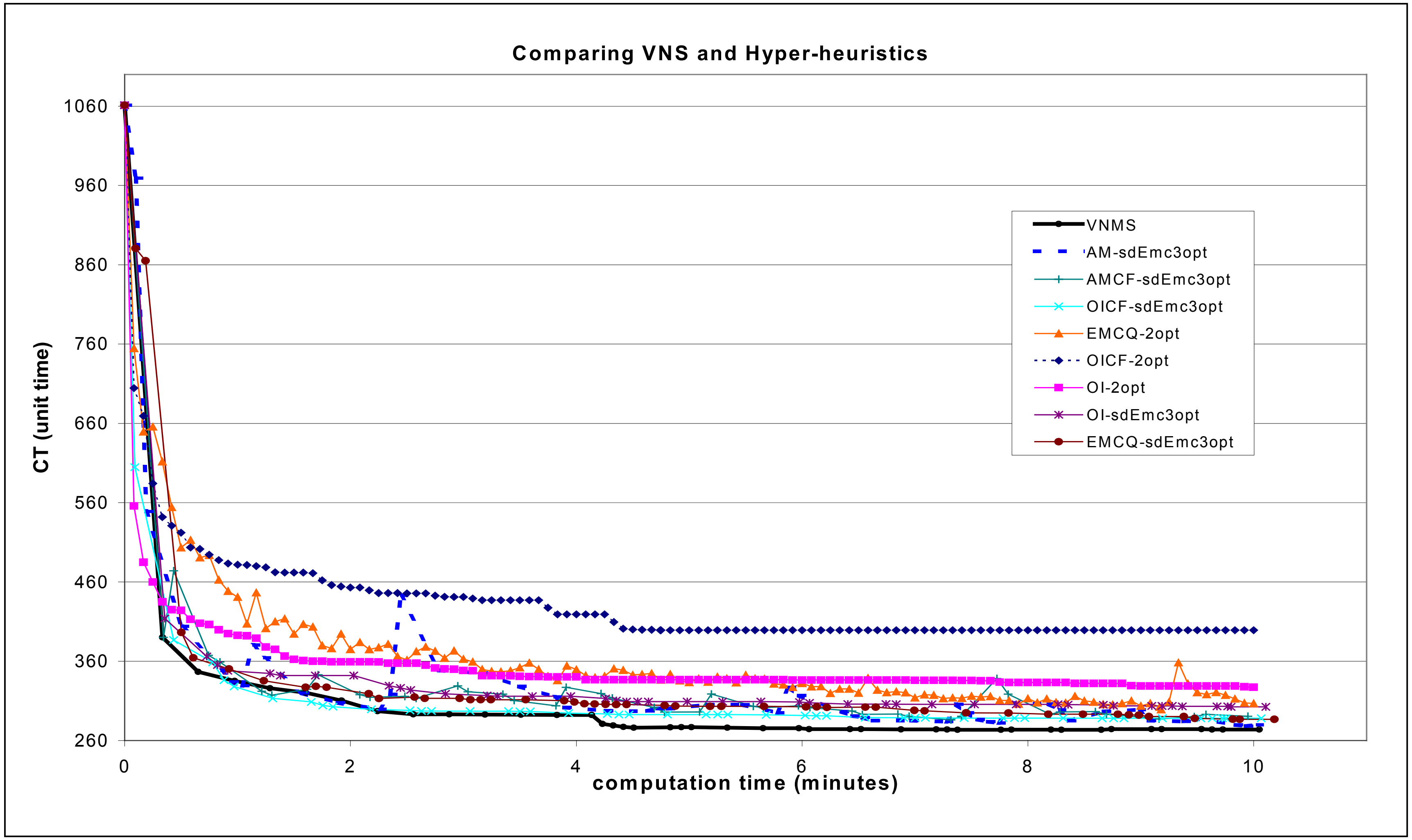


Figure 7.12: A comparison of VNMS and hyper-heuristics approaches.

Therefore, we further developed another Variable Neighbourhood Search with Exponential Monte Carlo (VNMS) acceptance criterion. The VNMS is a descent-ascent heuristic that operates on three sets of neighbourhood structures that are grouped together based on three different local search/operator approaches. Each group contains six neighbourhood structures. The first two sets use a steepest descent and EMC local search whilst the third set uses a random 3-opt operator. The solution returned by a local search, after exploring a neighbourhood structure, will be accepted based on the EMCQ acceptance criterion. The EMCQ acceptance criterion always accepts an improved solution. However, the probability of accepting a worse solution increases as δ decreases and the counter of consecutive none improvement iterations, Q, increases. VNMS begins by exploring the neighbourhood in the steepest descent group until no further improvement or some other termination criteria is met. Secondly, the EMC neighbourhood will be explored to divert the search direction by probabilistically accepting some worse solution. After exploring all neighbourhood structures in the EMC neighbourhood group and the search is trapped in a local optimum, then a shaking procedure is applied by randomly selecting a neighbour in random 3-opt neighbourhood structures using 3-opt operator. The trial solution will be evaluated by the EMCQ acceptance criteria. If it is not accepted, another trial will be performed until the solution is accepted or another termination criterion is met. Once the trial solution produced by the shaking procedure is accepted, the procedures are repeated (until the termination criterion is met). The solution obtained from the shaking procedure will be used as an incumbent solution. Results show that the VNMS is capable of producing good quality and stable results (for smaller dataset) in solving the component pick-and-place sequence of multi-head SMD placement machine. Therefore, the VNMS is more reliable compared to the hyper-heuristic approaches tested in this work. The proposed framework of VNMS might be suitable for solving other types of SMD placement machines or even other problem domains. However, the local searchers are problem specific.

The next chapter details an optimisation of the hybrid pick-and-place SMD placement machine, which is a new SMD placement machine. The chapter more closely mirrors the real-world, which has not been addressed by this chapter and the previous chapters.

Chapter 8

Optimising the Hybrid Pick-and-Place Surface Mount Device Placement Machine

8.1 Introduction

This chapter focuses on optimising the hybrid pick-and-place machine, which is a new surface mount device (SMD) placement machine. We propose a methodology for an on-line constructive heuristic to optimise the component pick-and-place operations. The machine begins a pickup and placement operation once the first printed circuit board (PCB) point pair is scheduled. While the machine is moving, picking or placing other components, the scheduler concurrently schedules the subsequent PCB points. Since a nozzle change operation is very expensive (it significantly adds to the overall assembly time), the constructive heuristic gives highest priority to minimising the number of nozzle changes.

As discussed in chapter 5, most previous work involves an offline scheduler that is ineffective if there is a change in the machine set-up such as a component being missing or misallocated from the feeder carrier. Hence, in this work we investigate an on-line scheduling approach using a greedy search that can concurrently generate a schedule for the subsequent PCB points using spare CPU time during pick-and-place operations.

This approach is different from our work in chapter 5, which only allowed the machine to begin a pickup and placement operation once a complete schedule was available, prepared an improved schedule for the subsequent PCB only (not for the current PCB being processed), ignored the nozzle changes operation (it was assumed that all components can be picked up by the same nozzle type) and modelled different types of machine specification. This work more closely mirrors the real-world, which previous work has not addressed. The work presented in this chapter has been disseminated as follows:

a) Ayob, M. and Kendall, G. (2004). A Nozzle Selection Heuristic to Optimise the Hybrid Pick and Place Machine. *Proceeding of the 2004 IEEE Conference on Cybernetics and Intelligent Systems (CIS 2004)*, Singapore, 1259-1264.

b) Ayob M. and Kendall G. (2005c). A Weighted Nozzle Rank Heuristic to Optimise the Hybrid Pick and Place Machine. Submitted to the *International Journal of Production Research*.

8.2 Hybrid Pick-and-Place Surface Mount Device Placement Machine

In this work we investigate the hybrid pick-and-place machine (specifically a new DIMA machine called Hybrid P&P HP-110). The Hybrid P&P is a type of multi-head placement machine (as classified in chapter 3). The machine has four fixed feeder carriers (mounted on the four sides of the machine), a fixed PCB table, two vision cameras, a tool bank, a trash bin and a positioning arm head that is equipped with two pipettes. Each pipette holds a nozzle that is used to grasp the components. Each feeder bank consists of several feeder slots where the component feeders are located. Several kinds of component feeders are available to handle the various types of component packaging; tape, sticks and trays. The feeders are used to provide the machine with a continuous supply of components. The tape feeder is an intelligent feeder that is equipped with a microprocessor that stores component data. The PCB table holds the PCB in a locked position during a pick-place operation. The head and arm (or sometimes called robot arm) is movable in the X-Y direction simultaneously. The nozzles (tools) are changed automatically, from a tool bank, as necessary. The tool bank contains thirteen slots that can hold twelve nozzles with one free slot for use during nozzle change operation.

The operation of the machine begins by concurrently loading a PCB into the machine, and reading the PCB and feeder setup information. Next, the head travels above the PCB to check the *fiducial* marks to ensure the proper positioning of the PCB. The pick-and-place operations are then started. Once completed, the PCB is moved out of the machine and the next PCB is loaded.

A sub tour (we refer to a sub tour to differentiate from an overall tour, which is an operation to place all the required components onto a single board) means an operation taken by the robot arm to pick up and place a number of components (depending on the number of nozzles per head, i.e. at most two components for this machine) in a single trip. A sub tour starts with the robot arm moving (from the latest placement point) in the X and Y direction concurrently to pickup the appropriate component(s) (one or two (at most)) from the feeder(s) (assuming that the head is already equipped with a correct nozzle, otherwise a nozzle change is required). Simultaneous pickups (*SP*) can happen if the distance between the two pickup points (of the same sub tour) comes within a user defined tolerance. Otherwise, the robot arm needs to move to the second pickup point to pick up the second component after picking up the first component. Next, the arm travels in the X and Y direction simultaneously and positions itself at the cameras for component recognition and alignment, if the component has to be recognised and aligned using camera (i.e. a vision's component recognition). If the left nozzle holds a small vision component and the right nozzle holds a large vision component, then a simultaneous vision (*SV*) can be done. That is the two components can be inspected simultaneously. Otherwise, the robot has to perform the two component visions sequentially, with the added overhead of the robot arm having to position the next pipette/nozzle at the larger vision camera and extra component recognition/alignment time. If a defective component is found, the robot moves to throw the rejected component into the trash bin (located near the camera). Next, the robot arm travels in the X and Y direction simultaneously and positions itself at the point where the first component will be mounted. Finally, the robot arm moves down (Z-direction), mounts the component on the board before returning to its original position (moves up) and repeats these steps for the next location on the board that have to be mounted on the same sub tour. After completing a sub tour, the robot arm returns to the feeder location to begin another sub tour (if a nozzle change is not required). If both (left and right) nozzle hold the mechanical alignment components (MA), the robot arm can move directly to X-Y placement position on the PCB after pickups the components from the feeders without having to perform camera recognition operation. The robot arm carries out an on the fly alignment for mechanical alignment component. That is the component is aligned while the robot arm is moving.

Figure 8.1 and figure 8.2 are a sketch diagram and a photograph of the HP-110 SMD placement machine.

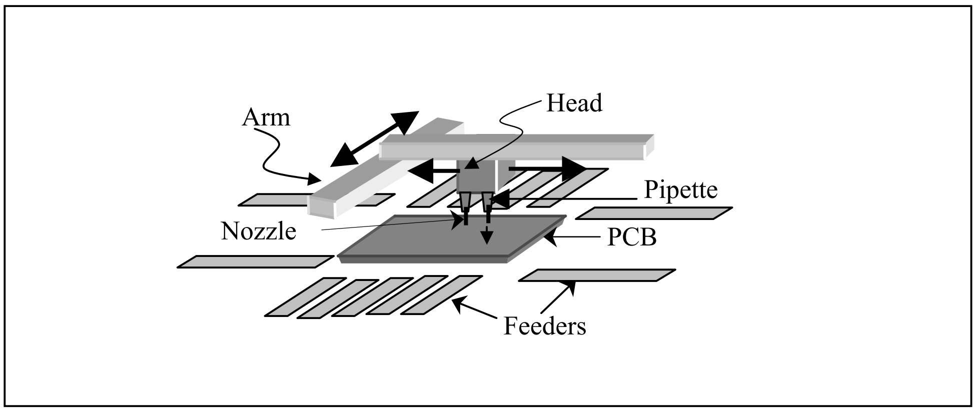


Figure 8.1: A sketch diagram of HP-110 SMD placement machine.

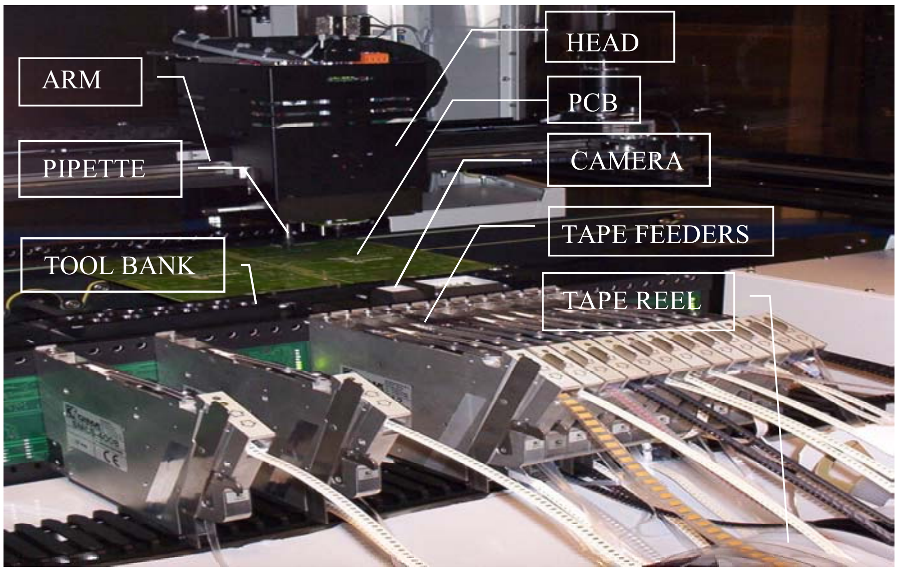


Figure 8.2: The Hybrid Pick-and-Place SMD placement machine.

8.3 Problem Statement and Assumptions

As the HP-110 has a single head equipped with two pipettes, which can hold two nozzles, a good selection of nozzle pair is important in order to minimise the number of nozzle changes to improve the efficiency of the machine. Moreover, the nozzle change operation is very time consuming (Crama et al., 1990; Jeevan et al., 2002; Magyar et al., 1999; Safai, 1996; Shih et al., 1996). For example, the HP-110 takes about two seconds for nozzle changeover. We assume that the total number of required nozzles is less than the capacity of tool bank (i.e. less than 13 in this specific case). We make this assumption, as there always has to be at least one spare slot in the tool bank to deposit a tool before picking up another one. This is a realistic assumption and, in fact, is adhered to on physical machines. We further assume that all components have the same priority and handling time (i.e. the time for pickup, placement and transportation from pickup point to placement point is the same for all components). This work addresses the scheduling problem for a single machine and a single board type and it focuses on the component placement sequencing with nozzle optimisation. So far, PCB machines vendor's software are not capable of solving even the single machine optimisation problem effectively (Magyar et al., 1999). In this work, we ignore the tray feeder problem since it poses a more challenging optimisation problem. A platform (that holds the trays component feeders) changeover takes about 10 seconds (for the HP-110 machine). Software vendors ignore this problem at the moment, assuming that there is a fixed set of tray feeders that are available throughout the scheduling process.

There are various types of component packaging and each packaging type is associated with a certain nozzle type. Each component packaging type can be associated with more than one nozzle type, and vice-versa. The problem is more complicated when one component type can have more than one type of packaging. This means that each PCB point on the board can be placed with more than one component packaging type. The component packaging type can be recognised and aligned without vision camera (i.e. using mechanical alignment on fly), using small vision camera and/or large vision camera, depending on the component packaging specification. As the small vision camera is located to the left of the large vision camera, then we can have a simultaneous vision and alignment operation (*SV*) if the left nozzle holds a small vision component and the right nozzle holds a large vision component. That is, the two components can be inspected simultaneously, which leads to time saving. It is more economical (in terms of assembly cycle time) to have both mechanical alignment components in the sub tour (MA) rather than having both vision components since the MA sub tour eliminates the time for moving to the camera and perform component recognition and alignment.

The two pipettes on the placement head are fixed at positions such that a simultaneous pickups (*SP*) operation can happen if the distance between the two pickup points (of the same sub tour) comes within a user's defined tolerance. The SP sub tour can also enhance the machine's throughput.

The feeder also takes a long time (i.e. about 0.5 second in this case) to transport a component from the component feeder to a pickup point. Therefore we should avoid picking up from the same component feeder in a sub tour.

Having four fixed feeder carriers (mounted on the four sides of the machine) also provides a great challenge in optimising the pick-and-place operations of this machine, since a pickup from the same feeder bank in a sub tour is better (in term of assembly cycle time) than a pickup from different feeder banks. It is apparent from the various situations, conditions and constraints described above, the optimising the assembly cycle time of this type of machine is a challenging scheduling problem.

8.4 The Scheduling Model

In this work, we propose an on-line scheduling approach and nozzle selection heuristics to optimise the HP-110. As the nozzle changeover operation is very time consuming, our approach aims to minimise the nozzle changes in order to minimise the total assembly cycle time (*CT*). The *CT* is the total time taken by the machine to assemble all the components onto a printed circuit board (PCB). Minimising the *CT* will directly increase the machine's throughput. The *CT* can be used to evaluate the quality of a schedule. The following notations are used to describe the scheduling model (some notational differences are used from those in chapters 4 and 5 due to the different machine types):

CT : the assembly cycle time to assemble all components;

B : the total number of sub tours;

λ : the time for picking up a component;

θ : the time for placing a component;

j : the j^{th} sub tour number where $j \in \{1,2,\ldots,B\}$;

$I(j)$: the time taken for the robot arm to travel from feeder to PCB point and place the component(s) in the j^{th} sub tour;

$P(j)$: the time taken for the robot arm to travel from PCB point to feeder and pick the component(s) in the j^{th} sub tour;

$\Phi_0(j)$: the time taken for the robot arm to move from PCB point to pickup the first component in the j^{th} sub tour from a feeder;

$\Phi_1(j)$: the time taken for the robot arm to move from current feeder to the next feeder in the j^{th} sub tour;

$b_0(j)$: the time taken for the robot arm to travel from the camera (or pickup point for the mechanical alignment case) to the first PCB point in the j^{th} sub tour;

$b_1(j)$: the time taken for the robot arm to travel from the first PCB point to the second PCB point in the j^{th} sub tour;

$C_0(j)$: the time taken for the robot arm to travel from feeder to position the first pipette above the camera in the j^{th} sub tour;

$C_1(j)$: the time taken for the robot arm to position the next pipette above the camera in the j^{th} sub tour;

$\rho(j)$: a decision variable either there is a second component for pickup and placement ($\rho(j)=1$) or 0 otherwise;

$\tau(j)$: a decision variable for having one camera vision and one mechanical alignment component in a sub tour where $\tau(j)=0$ if true, or 1 otherwise;

$\eta(j)$: the number of tool changes required to pickup the component(s) in the j^{th} sub tour where $\eta(j) \in \{0,1,2\}$;

$\gamma(j)$: a decision variable of simultaneous vision in the j^{th} sub tour where $\gamma(j)=0$

if exist simultaneous vision, or 1 otherwise;

$\omega(j)$ *: a decision variable of simultaneous pickup in the j^{th} sub tour where $\omega(j)=0$ if exist simultaneous pickup, or 1 otherwise;*

$\sigma(j)$ *: a decision variable of having two mechanical alignment components in the j^{th} sub tour where $\sigma(j)=0$ if having two mechanical alignment components, or 1 otherwise;*

u *: the time for the robot arm moves up/down;*

Ω *: the tool changing time;*

α *: the image acquisition and recognition time.*

ψ *: a decision variable either both components are picked up from the same component feeder in a sub tour ($\psi=1$), or $\psi=0$ otherwise;*

ζ *: the component feeder transportation time.*

The objective function is to:

$$\text{Minimise} \qquad CT = \sum_{j=1}^{B}\left[P(j)+I(j)\right] \tag{8.1}$$

Where:

$$P(j)=(\Phi_0(j)+\lambda+(2*u))+(\rho(j)*\omega(j)*(\max(*\Phi_1(j),\psi*\zeta)+\lambda+(2*u)))+(\eta(j)*\Omega) \tag{8.2}$$

$$I(j) = ((2*u)+ b_0(j)+ \theta+ \sigma(j)*(C_0(j)+\alpha)) + (\rho(j)*(\gamma(j)*\sigma(j)*\tau(j)*(C_1(j)+\alpha)+ b_1(j)+\theta+(2*u))) \tag{8.3}$$

The *CT* is computed by adding the total time taken by the robot arm to travel from the PCB point to the feeder(s) and picking up the component(s) (i.e. *P(j)*), and then travelling back to the PCB (i.e. the placement point) and placing the component(s) (i.e. *I(j)*). In this formulation we ignore the time of PCB loading, fiducial checking and PCB transfering (since these time are constant and only happen once for each board). We also ignore a simultaneous placement sub tour because the chances of having the simultaneous placement is very small due to the nature of the problem.

For each j^{th} sub tour, *P(j)* is a summation of the time taken for the robot arm to travel from a PCB point to the first pickup point (*$\Phi_0(j)$*), move down/up

(2*u), pickup the first component (λ); and if there is no simultaneous pickup and there is the second component to be picked up (i.e. $\rho(j)=1$ and $\omega(j)=1$), it includes the time taken for the robot arm to move from the current feeder to the next feeder ($\Phi_1(j)$), move down/up (2*u) and pick up the second component (λ). When both components need to be picked up from the same component feeder ($\psi=1$), then the time required to pick up the second component is usually dictated by the time taken by the feeder to transport the second component from the component feeder to the pickup position ($\psi*\zeta$) and $\Phi_1(j)$ factor is eliminated (since $\Phi_1(j)<\zeta$). If the j^{th} sub tour requires a nozzle change, then the $P(j)$ also includes the nozzle changing time ($\eta(j)*\Omega$).

The $I(j)$ for the j^{th} sub tour consists of the time taken for the robot arm to travel from the feeder to position the first pipette above the camera ($C_0(j)$), recognise the first component (α), travel from the camera to the first PCB point ($b_0(j)$), move down/up (2*u), and the time for placing the first component (θ); and if there is no simultaneous vision and there is the second component to be placed (i.e. $\gamma(j)=1$ *and* $\rho(j)=1$), it includes the time taken for the robot arm to position the next pipette above the camera ($C_1(j)$), to recognise the second component (α). If there is the second component to be placed, then the $I(j)$ also includes the time to travel from the first PCB point to the second PCB point ($b_1(j)$), move down/up (2*u), and the time for placing the second component (θ). However the recognition time of the second component and the time taken for the robot arm to position the next pipette above the camera can be eliminated if the machine can perform a simultaneous vision. The $C_0(j)$, $2*\alpha$ and $C_1(j)$ factors are eliminated when a sub tour involves two mechanical alignment components ($\sigma(j)=0$), whilst if a sub tour has one mechanical alignment and one vision components ($\tau(j)=0$), only $C_1(j)$ and α factors are eliminated. Therefore, the machine throughput can be increased by maximising the number of simultaneous vision sub tours, having both components mechanical's aligned in a sub tour, simultaneous pickup sub tour, minimising the number of nozzle changes and avoiding pickup from the same component feeder.

8.5 On-line Scheduling for Hybrid Pick-and-Place Machine

Based on our discussion with DIMA's machine expert, we found that an on-line scheduling approach is suitable to optimise the HP-110 due to a dynamic nature of the PCBA production process and the ability of the HP-110 to detect the exact availability and location of each component type on the feeder slot. In the PCBA production line, there are many spontaneous circumstances such as the component feeders are misallocated by the machine's operator, running out of components or defective components etc.

8.5.1 Data Requirements

The proposed on-line scheduling algorithm incorporates a database and an interrupt feature. We utilise a database that has eight tables; *PCBpoints, COMPONENT, FEEDER, NozzleComp, NOZZLE, NozzleGroup, MultiPickup* and *FinalSchedule*. Of course, any other data structure could be used to store this information but the following description is given in terms of a database and is presented to assist other researchers reproduce our results by having access to the data usage we employ.

The *PCBpoints* table contains information about the PCB points for the current board stored as *X-Y* coordinates, the component ID (identification), component type and the status of the PCB point either unscheduled, scheduled to be placed, being placed, placed and unavailable (the required component is missing).

The *COMPONENT* table stores the component data, this being a component type, a feeder type, the required number of feeder slots and a vision type (i.e. small or large vision – which component can be inspected by which type of camera is user defined). The *COMPONENT* table may have component types that are not applicable for the current PCB.

The *FEEDER* table contains data about the component's arrangement on the feeder slots. These are a component type, a slot number (where the component is located), a feeder bank name, a slot's distance and a counter. The 'counter' is automatically set when the machine feeder updates the database.

There is a counter, which tracks how many components are available on the feeder slot, taking into account those that are currently scheduled. This will allow the scheduler to identify the component type that will out run during pick-and-place operations and schedules the PCB points having available component types on the feeders.

The *NOZZLE* table stores a list of available nozzles and their location in the tool bank. All the nozzles that are suitable for the machine can be listed in the table. However, the nozzle types that are not located in the tool bank will have 'location'=0. The machine's user only needs to create the *NOZZLE* table records the first time the machine is used. Then, the system will automatically update the records based on the current availability of the nozzle type in the tool bank. The table is updated during machine initialisation procedure and will then automatically be updated using interrupt features if there is a change in the tool bank.

The *NozzleComp* table associates a component type to a nozzle type indicating, which nozzle(s) can be used to pickup each component. Each component type can be associated to more than one nozzle type (depending on the component's specification) and vice versa. The assignment of the component type to the nozzle type is given by the machine's user, but the system will advise the user if the assignment is not practical.

The *NozzleGroup* table is a temporary table containing the latest information about the nozzle pairs that can be used by the scheduler. It has a left nozzle ID, a right nozzle ID, a counter for simultaneous vision (*SVcounter*), a counter for same feeder bank pickup (*SFcounter*) and a counter for simultaneous pickup (*SPcounter*). *SVcounter*, *SFcounter* and *SPcounter* will be explained in section 8.5.2. The contents of the *NozzleGroup* table is always updated by the scheduler whenever there is a change in the *PCBpoints* table (i.e. after a component been scheduled or a new component needs to be scheduled due to a feeder being reloaded or a component has been found to be defective).

The *MultiPickup* table stores the current information about the component's pairs and nozzle pairs that are possible for simultaneous pickup.

The *FinalSchedule* table has a left component ID, a right component ID, a left nozzle ID, a right nozzle ID, a sub tour index, a sequence in a sub tour, a pickup style (*SP*: simultaneous pickup, *SV*: simultaneous vision, *SF*: same feeder bank pickup, *DF*: different feeder bank or *V*: single pickup) and a status (scheduled, being placed, placed, the required component is missing and the picked component is rejected, i.e. thrown into the trash bin). The left component ID and the right component ID are referring to the PCB point identification that will be picked up by the left or right nozzle, respectively. The sub tour index indicates the step when the sub tour will be performed, where the first index denotes as '0' and the k^{th} index is denoted as 'k'. A single tour pickup means that there is only one component needed to be picked up in the sub tour (in this case we only consider a vision component since a mechanical aligned component is ignored in this work). *DF* means that a component pair is picked up from different feeder banks. The *FinalSchedule* table will be used by the machine for guiding the picking and placing operations (i.e. the final schedule to sequence the pick-and-place operations). It provides information about which nozzle pairs to be used, the component pair to be picked up by each nozzle, the PCB points to receive the components and the sequence of pickups and placements in a sub tour. The *FinalSchedule* table is generated/updated by the scheduler or the appropriate ISR (interrupt service routine) every time when a new PCB point is scheduled, the component runs out or a feeder changeover occurs.

Immediately, when the machine is switched 'ON' and the PCB is loaded into the machine, the PCB points data is downloaded into the *PCBpoints* table and the intelligent feeders will update the content of the FEEDER table and PCBpoints table by activating the appropriate interrupt service routines (ISR) to allow the communication between the feeder's controllers and the central processing unit (CPU) of the machine. Therefore, the machine and scheduler know which components are missing, which components are available (having components on the feeder banks and how many components are available for each component type), the components that have to be scheduled, the location of PCB points, the exact location of each component on the feeder banks and the unused component types. Again, the ISR will be activated if a feeder changeover, feeder reloading, modification to a tool bank or a missing component occurs to ensure the correctness of the data in the FEEDER table, the PCBpoints table, the *NOZZLE* table and the *FinalSchedule* table. Since major machine parts are equipped with their own microprocessor (Dima, 2003), the machine can continuously operate with the ongoing process unless the machine needs to fetch data from the database for the next pick-and-place operations whenever a feeder changeover, feeder reloading or missing component occurs. The database is defined as a critical section to avoid a race condition. As such, only one process can update the database at a given time.

If the machine does not have a schedule for the PCB being processed, then our on-line scheduler will immediately construct the schedule once the feeders have completed updating the database. Based on the records from the database, the scheduler will only schedule the PCB points that have components available on the feeders and need to be scheduled. A detailed explanation as to how the PCB points are selected for scheduling can be found in section 8.5.2. The machine begins the pickup-and-placement operations once the first component pair is scheduled. While the robot arm is moving, picking up or placing component, or recognising/aligning the component; the scheduler may employ the CPU free time to schedule the subsequent PCB points. This could be done since major machine parts are equipped with their own microprocessor (Dima, 2003), i.e. they can work independently. Every time, after a PCB point is scheduled, the scheduler will update the *'status'* in the *PCBpoints* table to *'scheduled'*, it adds a new record into the *FinalSchedule* table with *status='scheduled'* and decreases the available component counter.

If a defective component is detected (after being picked up), then the appropriate PCB point will be rescheduled. This can be done by activating another ISR (activated by the robot arm) to reset the status of the appropriate PCB point in the *PCBpoints* and *FinalSchedule* tables.

If a missing component event occurs after the PCB points that is expecting the component type have been scheduled (or waiting to be scheduled), then the feeder's controller will activate the ISR to update the database such that the appropriate PCB points will be unscheduled (or will not be scheduled) by changing the status (in the *PCBpoints* and *FinalSchedule* tables) from *scheduled* to *'unavailable'*. The PCB points that are paired with the missing components PCB points, will also be rescheduled to improve the schedule. This only affects the PCB points that have been scheduled and waiting for picking and placing. If the missing/rejected component, or reloading component, happens after the scheduler has completed generating the schedule, then the feeder's controller/robot arm will activate the appropriate ISR to update the database such that the appropriate PCB points will be unscheduled (or will not be scheduled) and/or reschedule the appropriate PCB points that have components available on the feeder bank (that are not yet been scheduled due to the missing component). The feeder's controller also updates the FEEDER table when the feeder reloading, feeder changeover or component run out event occurs (by activating the appropriate ISR).

The decision whether to proceed with the next PCB by moving out the uncompleted current PCB due to missing components' type or waiting until all the missing components have been reloaded and completing the current PCB, is dependent on managerial policy. However, the machine is intelligent enough to detect this situation.

8.5.2 On-line Constructive Heuristics

In this section (section 8.5.2.1 and 8.5.2.2), we develop two constructive heuristics. The difference between the two heuristics are in the nozzle ranking procedure (the way of counting the SV sub tours) and the importance of SV and SF sub tour operations.

We denote *SP* as a simultaneous pickup, *SV* as a simultaneous vision, *SF* as a same feeder bank pickup, *SC* as a same component feeder pickup, *DF* as a different feeder bank pickup, *MA* as having two mechanical alignment components, *MV* as having one mechanical and one vision component, *M* as having only one mechanical alignment in a sub tour and *V* as a one vision component in a sub tour (i.e. for *M* and *V*, there is only one component in a sub tour).

Let *P* be a set of available component packages on the feeder bank that are required by the current PCB, a component package $p \in P$, *T* is a set of available nozzle, *C* is a set of PCB points that have to be scheduled (having components available on the feeder bank) and *Q* is a sum of PCB points that have component's available on the feeder bank and need to be scheduled. Q is decreased once the PCB point is scheduled.

Each nozzle pair is associated with some counters that count the number of *MA* sub tours (*MAcounter*), *SP* sub tours (*SPcounter*), *M* sub tours (*Mcounter*), *MV* sub tours (*MVcounter*), *SV* sub tours (*SVcounter*), *SF* sub tours (*SFcounter*), two components sub tours (*STcounter*) and one component sub tours (*Scounter*). These counters only count based on the availability of PCB points that need to be scheduled, which have the component packages available on the feeder and have not been scheduled yet.

Z is a decreasing ordered list of available nozzle pairs. There are three nozzle ranking procedures, these being an R_0 and the two weighted nozzle rank procedures, R_1 and R_2 that are discussed in section 8.6. Z is generated using one of these procedures depending on which heuristic is to be applied. $z \in Z$ is the z^{th} nozzle pair where $z \in \{0,1,2,..D\}$ and D is a sum of available nozzle pairs. The z^{th} nozzle pair has a left nozzle ($L_z \in T$) and a right nozzle ($R_z \in T$). The $p_{Lz} \in P$ and $p_{Rz} \in P$ are the component packages that can be picked up by L_z and R_z nozzle, respectively. $c_{pLz} \in C$ and $c_{pRz} \in C$ are a set of PCB points that are expecting the p^{th} component package to be placed there by the z^{th} nozzle pair, left and right, respectively. All the counters for the z^{th} nozzle pair are denoted with *z, such as MAcounter(z), SPcounter(z),* etc.

8.5.2.1 On-line Constructive Heuristic A

An on-line constructive heuristic A and an on-line constructive heuristic B are our priliminary works to optimise the HP-110. Based on our early discussion with DIMA, we have proposed these two heuristics. As this is a preliminary work, there are some issues that have been overlooked in this section. These are:

a) The nozzles should be associated with the component packages. A component type can have more than one type of packaging. In these sections (section 8.5.2.1 and 8.5.2.2) the nozzles are directly associated with the component type, which is wrong.

b) The component recognition and alignment is a user defined. In these sections (section 8.5.2.1 and 8.5.2.2), we assume the small component can be visualised by both the small or large vision cameras, which is wrong.

c) Some component packaging type can be aligned while the robot arm is moving that is using mechanical alignment on fly (without using vision camera). In these sections (section 8.5.2.1 and 8.5.2.2), we do not consider the mechanical alignment component.

d) The feeder takes a long time to transport a component from the component feeder to a pickup point. This was ignored in these sections (section 8.5.2.1 and 8.5.2.2).

Since the problem definition for these works (section 8.5.2.1 and 8.5.2.2) is slightly difference from the rest of the works due to overlooked issues, we use slightly different notations. Let K be a set of available component types on the feeder bank that are required by the current PCB, a component type $k \in K$, c_k is a set of PCB points that are expecting the k^{th} component type to be placed there where $c_k \in C$.

Z is a decreasing ordered list of available nozzle pairs based on the maximum of *SVcounter*, then *SPcounter* and *SFcounter*.

The constructive heuristic A (with nozzle re-optimisation) is shown in figure 8.3. The algorithm starts by counting the number of *SV*, *SF* and *SP* sub tours that can be performed by each nozzle pair. It sorts the PCB points starting with the minimum of maximum (X,Y) coordinate (then with the minimum (X,Y) when duplication of maximum (X,Y) exists). This approach was introduced in chapter 6 section 6.3.1.

In this heuristic, we count a *SV* if the left nozzle or both nozzles pick up a small vision component. The first top nozzle pair, $z_0 \in Z$ is chosen for the next pickups and placements. If there exist component(s) that need to be scheduled ($Q>0$), then the algorithm tries to schedule pairs of components that are possible for simultaneous pickups by the current nozzle's pair (i.e. if $SPcounter > 0$). We decided to schedule the simultaneous pickup sub tours first instead of simultaneous vision sub tours because we can obtain a better time saving (refer to table 8.5 and 8.10). Indeed, if the simultaneous vision sub tours are scheduled first, then it may not allow the simultaneous pickup sub tours. Without changing the nozzle, then we choose pairs of components that allow simultaneous vision with a same feeder bank pickup, *SV+SF* (i.e. *SVcounter>0* and *SFcounter>0*). Next, we select pairs of components that only allow same feeder pickups. That is, when the *SF>0* and *SV=0*. When there are no more appropriate components that allow an *SF* pickup, then we schedule a different feeder banks (*DF*) sub tours (i.e. *SF=0* Step 3.1.3 in figure 8.3). However, we try to improve the total assembly cycle time by creating a simultaneous vision whenever possible.

```
1. Sort the PCB points as in section 6.3.1.
2. Create nozzle pair list, Z and choose $z_0 \in Z$.
3. If Q>0;
  REPEAT
    3.1 Using the same nozzle pair:
        3.1.1 If SPcounter > 0 Then
              Schedule for SP sub tours by choosing some $k \in K$ and then
              $c_k \in C$;
        3.1.2 If SFcounter > 0 Then
              a) Schedule for SV with SF sub tours by choosing some $k \in K$
                 and then $c_k \in C$;
              b) Schedule for SF sub tours by choosing some $k \in K$ and then
                 $c_k \in C$.
        3.1.3 Otherwise (i.e. SF=0)
              Schedule for DF sub tours by choosing some $k \in K$ and then
              $c_k \in C$;
    3.2 Nozzle changing:
        If Q>0 then re-generate the Z list and choose the best nozzle pair,
        $z_i \in Z$ by changing only one of the nozzle if possible, or otherwise
        change both nozzles.
  UNTIL Q=0.
3. Re-optimise the nozzle changes.
```

Figure 8.3: An adaptive constructive heuristic A algorithm.

For each sub tour, the placement points (PCB points) are selected once the pickup components are scheduled. The scheduler knows which component(s) are to be scheduled for picking up (avoiding unnecessary pickups) by referring to the database. If there exists more than two PCB points that can receive the components being assigned to the nozzles, then we use a nearest neighbour heuristic to select between the two PCB points. We choose the appropriate pair of PCB points that are close to each other (using a chebychev distance i.e. we consider a minimum of maximum *(|X1-X2|, |Y1-Y2|)* where *X1, Y1* and *X2, Y2* are the *X, Y* coordinates of both PCB points). Once the PCB point is scheduled, its status in the database is changed to ‘scheduled’ to avoid it being scheduled again. However, the PCB point’s status can be changed back to ‘unscheduled’ if the machine cannot place the component onto the PCB due to the component being defective. This forces the affected PCB point to be rescheduled later. The task to reschedule the defective component types can be handled by an appropriate ISR.

After scheduling all the available PCB points that should be picked up by the current nozzle (except the pair of PCB points that need to be picked up from a different feeder bank i.e. Step 3.1.1 and 3.1.2 in figure 8.3), we need to change the nozzles. In order to minimise the number of nozzle changes, since nozzle changes take a long time compared to other operations (such as vision, picking, moving and placing), we try to change only one nozzle at a time and choose the best nozzle pairs to be applied in the next cycle, which has maximum *SV*, *SP* and *SF* sub tours.

If there are no more component pairs from the same feeder bank (*SF*=0), then we apply the following procedure to choose the best nozzle pair in order to minimise the assembly cycle time. If the component pair can be picked up by the current nozzle pair, then we maintain the nozzle pair. Otherwise, we try to reuse the previous nozzle pairs by preferring the pair that allows simultaneous vision. If the previous nozzle pairs are not applicable, then we look for a nozzle pair by changing only one nozzle (trying to have simultaneous vision if possible) or otherwise change both nozzles (also trying to have simultaneous vision if possible).

Finally, when all the available PCB points, having components available on the feeder slots, have been scheduled, we reoptimise the schedule by minimising the nozzles changes. To reoptimise the nozzle change, we begin by sorting the schedule based on the sub tour index in ascending order, but only consider the unplaced component pairs. We create a list of unique nozzle pairs. Each nozzle pair is associated with a counter (that counts the number of sub tours for the nozzle pair) and the starts of a sub tour index. If there exists a nozzle pair duplication, then we sum the counters, eliminate one of them and adjust the sub tour index of the other pairs appropriately. We try to ensure (as far as possible) that the nozzle change only happens whenever there are no more components that can be picked up by the current nozzle. If we must do a nozzle change, then we try to change only one nozzle if possible.

The 'final schedule' that has been generated by the scheduler is stored in the *PCBpoint* database in *FinalSchedule* table.

8.5.2.2 On-line Constructive Heuristic B

In order to increase the possibility of picking up a small vision component by the left nozzle and reducing the possibility of picking up the small vision component by right nozzle, we only count a simultaneous vision if the left nozzle picks up a small vision component and the right nozzle picks up a large vision component. However, once the sub tour is scheduled, the simultaneous vision occurs when both cameras can perform simultaneous vision/recognition on both components. The constructive heuristic B (with nozzle re-optimisation) is shown in figure 8.4.

Similarly, the nozzle pairs are ranked starting from the maximum *SVcounter*, then maximum *SPcounter* and *SFcounter*. The first nozzle pair, $z_0 \in Z$ is chosen for the next pickup and placement operations. Once the nozzle pair is selected, we then schedule the component pairs and PCB points for simultaneous pickups (if possible). With the same nozzle pair, we then schedule pairs of components and PCB points that allow a simultaneous vision with a same feeder bank pickup (Step 3.1.2a in figure 8.4 i.e. $SV>0$ and $SF>0$). Next, we select component pairs and PCB points that allow the simultaneous vision to occur i.e. $SV+DF$ sub tour (focusing more on SV sub tour compared to SF sub tour i.e. Step 3.1.2b in figure 8.4). Once the same nozzle pair cannot make any more simultaneous vision, then we schedule for same feeder bank pickup (Step 3.1.2c in figure 8.4).

```
1. Sort the PCB points as in section 6.3.1.
2. Create nozzle pair list, Z and choose $z_0 \in Z$.
3. If $Q>0$;
  REPEAT
   3.1 Using the same nozzle pair:
      3.1.1 If SPcounter > 0 Then
             Schedule for SP sub tours by choosing some $k \in K$ and then
             $c_k \in C$;
      3.1.2 If SVcounter > 0 or SFcounter > 0 Then
             a) Schedule for SV with SF sub tours by choosing some $k \in K$
                and then $c_k \in C$;
             b) Schedule for SV sub tours by choosing some $k \in K$ and then
                $c_k \in C$;
             c) Schedule for SF sub tours by choosing some $k \in K$ and then
                $c_k \in C$;
      3.1.3 Otherwise (i.e. $SV=0$ and $SF=0$)
             Schedule for DF sub tours by choosing some $k \in K$ and then
             $c_k \in C$;
   3.2 Nozzle changing:
      If $Q>0$ and $S(z_i)=0$ then re-generate the Z list and choose the best
      nozzle pair, $z_i \in Z$ by changing only one of the nozzle if possible, or
      otherwise change both nozzles.
  UNTIL $Q=0$.
3. Re-optimise the nozzle change.
```

Figure 8.4: An adaptive constructive heuristic B algorithm.

If the selected nozzle cannot be scheduled for simultaneous vision or same feeder bank pickup sub tour (i.e. *SV*=0 and *SF*=0), we schedule a different feeder (*DF*) sub tour pickup. In this case there is no possibility of having simultaneous vision sub tours even with a different feeder bank pickups.

For each sub tour, the placement points (PCB points) are selected once the pickup components are scheduled (same as the procedure in the constructive heuristic A). The rest of the algorithm is the same as the constructive heuristic A.

8.5.3 Testing and Results

In this work, we use the average machine operation time given by DIMA to estimate the *CT* as an evaluation for our heuristic performance. This is different from our evaluation function used in chapter 4, 5, 6 and 7 (in which we only considered minimising the robot travelling distance and/or feeder carrier and PCB table movement). In fact, to date, none of work in this field uses the average machine operation time to evaluate the machine throughput. Many researchers are only concerned with minimising the robot travelling distance (and/or feeder carrier and PCB table movement) in order to improve the machine throughput (or particularly, the component pick-and-place sequence). Basically, the *CT* of many SMD placement machine types is dependent on many factors such as nozzle changes (which is very time consuming), simultaneous pickup, simultaneous vision etc. (these factors are very machine dependent). Ignoring these factors in solving component pick-and-place sequencing might not be a good strategy. For example, solving the component pick-and-place sequencing by minimising the robot travelling distance without considering the nozzle change operation might incur many unnecessary nozzle changes, which is very inefficient. Of course, they might be able to produce a good quality solution. However, they may obtain a much better solution if the other crucial factors were also considered. Moreover, as the speed of robot arm (i.e. the arm and head) of the latest machines is very fast and the component density on the PCB is increased (i.e. the distance among PCB points tends to be smaller), minimising the robot travelling distance is becoming a less significant factor for improving the machine throughput. Indeed, due the acceleration/deceleration rate of the robot arm, the time taken for the robot arm to move short or longer distances might be fairly equal. Therefore, it is ineffective to just minimise the robot travelling distance in order to improve the machine throughput. For the purpose of optimising the component pick-and-place operation, exact information about the machine speed, acceleration/deceleration rate etc. is not necessary (as the machine is embedded with a control software for accurate movements/operations). The average machine operation time is adequate in guiding the search for a better quality schedule. Moreover, including the machine speed, acceleration/deceleration rate etc. might introduce a more complex formulation for the objective function.

We cannot compare our results with any other work since this is a new machine, a new problem and no previous work has been done to optimise the

HP-110. The average operation times are shown in table 8.1. To demonstrate our approach we generate a random dataset (dataset A). The PCB points, component specifications, nozzle specifications and feeder arrangement are randomly created. Dataset A contains 30 PCB points, 10 component types, 7 nozzle types and 2 feeder banks. In this work we set a user's defined tolerance as 45mm (user defined tolerance=nozzle gap).

Table 8.2 shows the arrangement of each component type on the feeder bank ('slot number', 'component type' and 'feeder bank'), the 'distance' in millimeter (with reference to the slot 0 pickup point of each feeder bank) and the 'camera vision' (indicating the camera to be used for the component vision/recognition either 'small' or 'large' camera). The 'distance' in table 8.2 indicates the distance of the component's pickup point from the slot 0 pickup point of the feeder bank. The 'distance' can be automatically computed based on the tolerance given by the user and the machine feeder slot's distance.

TABLE 8.1: THE AVERAGE PROCESSING TIME OF THE HP-110

Operation	**Time(ms)**
Pickup (λ)	10
Placement (θ)	10
Axis up/down (u)	50
Move to XY feeder ($\Phi_0(j)$)	350
Move to XY next feeder ($\Phi_1(j)$)	290/350$^+$
Move XY to camera ($C_0(j)$)	350
Move next pipette to camera ($C_1(j)$)	225
Image acquisition and recognition (α)	175
Move to XY place ($b_0(j)$)	300/410*
Move to XY next place ($b_1(j)$)	175
Tool changing (Ω)	2000
The component feeder transportation(ζ)	500

Note: * *for MA sub tour ;* $^+$*for DF sub tour.*

TABLE 8.2: THE FEEDER ARRANGEMENT AND SPECIFICATION

Slot number	Component type/Camera vision	Distance (mm)	Feeder bank
0	5/Small	0	A
3	6/Large	45	A
6	4/Large	90	A
9	8/Large	135	A
12	1/ Small	180	A
15	7/Large	225	A
18	9/Large	270	A
100	2/ Small	0	B
103	3/Large	45	B
106	10/ Small	90	B

Table 8.3 shows the assignment of each nozzle to the component types. Each nozzle can be used for picking up more than one component type and each component type can be picked up by more than one nozzle type. The decision of which nozzle type can be used for picking up the component is made by the machine user (with system guidance).

TABLE 8.3: THE ASSOCIATION OF COMPONENT TYPE AND NOZZLE

Nozzle ID	Component types that can be picked up by the nozzle
HP2703	2,5,1
HP2706	4,1,10
SMCS2705	7
SMCS2710	3,4,6,8
SMCS2715	3,6,8
SMCS2720	9

Table 8.4 contains the PCB point data; being ‘component ID’, ‘component type’ and the *‘X’, ‘Y’* coordinates of the PCB point.

TABLE 8.4: THE PCB POINTS SPECIFICATION

Component ID	Component type	X (mm)	Y (mm)
12	10	15.4	50.6
25	9	24.6	49.6
3	3	26.0	26.5
26	8	28.2	95.7
18	10	30.0	21.0
2	3	30.5	16.9
1	2	32.0	11.0
5	4	32.4	35.0
10	1	35.5	29.0
7	1	35.5	45.8
24	3	38.0	92.2
30	5	38.6	94.8
13	7	40.0	20.0
11	1	47.6	27.4
23	7	48.0	74.8
15	9	50.0	40.4
19	10	50.8	36.2
8	2	52.3	37.0
9	4	54.9	25.8
29	6	56.2	83.9
27	4	58.8	110.6
16	6	60.2	70.1
17	8	70.8	80.3
4	5	71.0	19.0
14	2	74.6	30.8
6	5	78.9	26.2
22	10	84.0	60.6
28	9	84.8	102.1
20	2	90.1	20.8
21	5	110.6	60.5

Note: The X,Y indicates the X,Y coordinates of the PCB point.

Based on the average processing time of the HP-110 and the time taken for one sub tour, we calculate the machine throughput in components per hour (cph) for each pickup and placement operation type. Table 8.5 summarises the machine throughput without a nozzle change operation based on one or two components pickup and placement operations. Table 8.5 shows the effectiveness of each pickup and placement operation type. The most efficient pickup and placement operation is to maximise the number of pickups and placements that has both simultaneous pickup and vision (SP + SV). A single pickup in a sub tour should be avoided since it is the worst pickup and placement operation unless another operation increases the number of nozzle changes. It is not worth increasing the number of nozzle changes just to have only one sub tour that has SP with SV since a sub tour that has SP with SV that incurs a nozzle change can cost about 3680 ms, which is even worse than just picking up one component per sub tour (1395*2 i.e. 2790 ms for picking up and placing two components). The SP sub tour consumes an equal amount of processing time as a sub tour of SV with SF. On the other hand, SV is better than SF and SF is better than DF. In conclusion, we can say that table 8.5 gives a significant clue as to how to devise a good strategy in constructing a good pickup and placement schedule (a complete summary is shown in table 8.10 that includes a mechanical aligned component and same component feeder pickups).

TABLE 8.5: THE THROUGHPUT OF THE HP-110.

Pickup and placement operation type	**Time (ms)**	**cph**
One component per sub tour (V)	1395	2580
SV+SP	1680	4285
SP	2080	3461
SV + SF	2080	3461
SV+DF	2140	3364
SF	2480	2903
DF	2540	2834

Note: Time in milliseconds per sub tour.

Table 8.6 and table 8.8 show an example of the result obtained by constructive heuristics A and B, before applying the nozzle change reoptimisation. Table 8.7 and table 8.9 show an improved schedule (based on the schedule in table 8.6 and table 8.8, respectively) after reoptimising the nozzle changes. The schedule in table 8.6 caused five nozzle changes with

CT=39260ms (for assembling 30 components) and producing 2750cph whilst the schedule obtained after reoptimising the nozzle change (result in table 8.7) only caused four nozzle changes with *CT*=37260ms and producing 2898cph. Results show that reducing by just one nozzle change can increase the machine's throughput from 2750cph to 2898cph (5.38% improvement). This demonstrates that minimising nozzle changes can increase the machine throughput.

The schedule in table 8.8 caused four nozzle changes with *CT*=37500ms (for assembling 30 components) and produced 2880cph, whilst the schedule obtained after reoptimising the nozzle change (table 8.9) only caused three nozzle changes with *CT*=35500ms and produced 3042cph. Results show that the constructive heuristic B with nozzle reoptimisation heuristic can increase the machine's throughput from 2750cph (for constructive heuristic A without nozzle reoptimisation heuristic) to 3042cph (10.62% improvement).

The result in table 8.7 shows that the constructive heuristic A produces some bad sub tours. These being: the 10th and 11th sub tour where both nozzles are assigned to pickup the small vision components. Consequently, there is a limited number of small vision components to be scheduled for picking up by the left nozzle of the subsequent sub tours. Therefore, sub tours 13 and 14 have to be scheduled for picking up all the large vision components (which does not allow simultaneous vision). On the other hand, table 8.9, shows that the constructive heuristic B is capable of producing a slightly better quality schedule compared to a schedule generated by the constructive heuristic A. Due to a decision that we count simultaneous vision only if a small vision component is assigned for the left nozzle pickup and a large vision component is assigned for the right nozzle pickup, and we try to maximise the simultaneous vision sub tour; then we obtain all the simultaneous vision sub tours (schedule in table 8.9). Indeed we got less nozzle change in table 8.9 (i.e. 3) compared to the schedule in table 8.7 (i.e. 4).

TABLE 8.6: THE EXAMPLE OF RESULT OBTAINED BY THE CONSTRUCTIVE HEURISTIC A (WITHOUT REOPTIMISE THE NOZZLE CHANGE)

Left component ID	**Right component ID**	**Left nozzle**	**Right nozzle**	**Sub tour number**	**Sequence**	**Status**	**Operation type**	***η(j)***
30	29	HP2703	SMCS2710	0	0	0	SV+SP	0
21	16	HP2703	SMCS2710	1	0	0	SV+SP	0
8	24	HP2703	SMCS2710	2	0	0	SV+SP	0
14	3	HP2703	SMCS2710	3	0	0	SV+SP	0
1	2	HP2703	SMCS2710	4	0	0	SV+SP	0
4	27	HP2703	SMCS2710	5	0	0	SV+SF	0
6	9	HP2703	SMCS2710	6	0	0	SV+SF	0
10	5	HP2703	SMCS2710	7	0	0	SV+SF	0
11	17	HP2703	SMCS2710	8	0	0	SV+SF	0
7	26	HP2703	SMCS2710	9	0	0	SV+SF	0
20	18	HP2703	HP2706	10	0	0	SV+SF	1
12	22	HP2706	HP2706	11	0	0	SV+SF	1
13	25	SMCS2705	SMCS2720	12	0	0	SP	2
23	15	SMCS2705	SMCS2720	13	0	0	SP	0
19	28	HP2706	SMCS2720	14	0	0	SV+DF	1

TABLE 8.7: THE EXAMPLE OF RESULT OBTAINED BY THE CONSTRUCTIVE HEURISTIC A AFTER REOPTIMISING THE NOZZLE CHANGE.

Left component ID	**Right component ID**	**Left nozzle**	**Right nozzle**	**Sub tour number**	**Sequence**	**Status**	**Operation type**	$\eta(j)$
30	29	HP2703	SMCS2710	0	0	0	SV+SP	0
21	16	HP2703	SMCS2710	1	0	0	SV+SP	0
8	24	HP2703	SMCS2710	2	0	0	SV+SP	0
14	3	HP2703	SMCS2710	3	0	0	SV+SP	0
1	2	HP2703	SMCS2710	4	0	0	SV+SP	0
4	27	HP2703	SMCS2710	5	0	0	SV+SF	0
6	9	HP2703	SMCS2710	6	0	0	SV+SF	0
10	5	HP2703	SMCS2710	7	0	0	SV+SF	0
11	17	HP2703	SMCS2710	8	0	0	SV+SF	0
7	26	HP2703	SMCS2710	9	0	0	SV+SF	0
20	18	HP2703	HP2706	10	0	0	SV+SF	1
12	22	HP2706	HP2706	11	0	0	SV+SF	1
19	28	HP2706	SMCS2720	12	0	0	SV+DF	1
13	25	SMCS2705	SMCS2720	13	0	0	SP	1
23	15	SMCS2705	SMCS2720	14	0	0	SP	0

TABLE 8.8: THE EXAMPLE OF RESULT OBTAINED BY THE CONSTRUCTIVE HEURISTIC B (WITHOUT REOPTIMISE THE NOZZLE CHANGE)

Left component ID	**Right component ID**	**Left nozzle**	**Right nozzle**	**Sub tour number**	**Sequence**	**Status**	**Operation type**	$\eta(j)$
30	29	HP2703	SMCS2710	0	0	0	SV+SP	0
21	16	HP2703	SMCS2710	1	0	0	SV+SP	0
8	24	HP2703	SMCS2710	2	0	0	SV+SP	0
14	3	HP2703	SMCS2710	3	0	0	SV+SP	0
1	2	HP2703	SMCS2710	4	0	0	SV+SP	0
4	27	HP2703	SMCS2710	5	0	0	SV+SF	0
6	9	HP2703	SMCS2710	6	0	0	SV+SF	0
10	5	HP2703	SMCS2710	7	0	0	SV+SF	0
11	17	HP2703	SMCS2710	8	0	0	SV+SF	0
7	26	HP2703	SMCS2710	9	0	0	SV+SF	0
12	25	HP2706	SMCS2720	10	0	0	SV+DF	2
22	28	HP2706	SMCS2720	11	0	0	SV+DF	0
19	15	HP2706	SMCS2720	12	0	0	SV+DF	0
18	13	HP2706	SMCS2705	13	0	0	SV+DF	1
20	23	HP2703	SMCS2705	14	0	0	SV+DF	1

TABLE 8.9: THE EXAMPLE OF RESULT OBTAINED BY THE CONSTRUCTIVE HEURISTIC B AFTER REOPTIMISING THE NOZZLE CHANGE

Left component ID	**Right component ID**	**Left nozzle**	**Right nozzle**	**Sub tour number**	**Sequence**	**Status**	**Operation type**	$\eta(j)$
30	29	HP2703	SMCS2710	0	0	0	SV+SP	0
21	16	HP2703	SMCS2710	1	0	0	SV+SP	0
8	24	HP2703	SMCS2710	2	0	0	SV+SP	0
14	3	HP2703	SMCS2710	3	0	0	SV+SP	0
1	2	HP2703	SMCS2710	4	0	0	SV+SP	0
4	27	HP2703	SMCS2710	5	0	0	SV+SF	0
6	9	HP2703	SMCS2710	6	0	0	SV+SF	0
10	5	HP2703	SMCS2710	7	0	0	SV+SF	0
11	17	HP2703	SMCS2710	8	0	0	SV+SF	0
7	26	HP2703	SMCS2710	9	0	0	SV+SF	0
20	23	HP2703	SMCS2705	10	0	0	SV+DF	1
18	13	HP2706	SMCS2705	11	0	0	SV+DF	1
12	25	HP2706	SMCS2720	12	0	0	SV+DF	1
22	28	HP2706	SMCS2720	13	0	0	SV+DF	0
19	15	HP2706	SMCS2720	14	0	0	SV+DF	0

Based on our experiments on a Pentium 4, 1.5Ghz, 256 MB RAM computer, we obtained the first component pair for the schedule in 6.39 seconds after the PCB data was downloaded into the machine and the following pair is ready about 1 millisecond later. While the PCB is being loaded onto the machine, the scheduler can use the CPU free time (since the CPU is in an idle state at this time) to generate the schedule. Therefore the machine can start the pickup and placement operation at 6.39 seconds after the PCB data is downloaded into the machine or, perhaps, immediately after the board (i.e. PCB) is ready (clamped on the PCB table) if the board loading time is more than 6.39 seconds. Since the schedule of the second component pair is ready less than 1ms later and the fastest time taken by the machine to complete a sub tour is more than 1 ms (i.e. a sub tour for MA+SP, see table 8.10), we can ignore the time taken to calculate the next run. Therefore, the machine can continuously run without any interruption once the schedule of the first component pair is ready. After the scheduler has completed generating the schedule (in about 14.75 seconds for 30 components), it will reoptimise the nozzle changes in order to improve the schedule. The nozzle reoptimisation procedure only takes about 110ms (for a 30 component schedule).

In this work we have focused more on minimising the nozzle changes, optimising the pickup and optimising the component's vision/recognition rather than optimising the placement operation. This is due to the fact that the distance among PCB points is relatively small (in general) compared to the distance of component's location on the feeder bank. Therefore we assume that optimising the placement operation is less important than minimising the nozzle changes, optimising the pickup and optimising the component's vision/recognition. However, we still agree with other researchers (such as Jeevan et al., 2002 and Ho and Ji, 2003 and 2004) that optimising the placement operation could lead to further maximising the machine throughput. Therefore, we also try to optimise the placement operation after optimising the nozzle changes, pickup operation and component's vision/recognition. This is done by choosing the pair of PCB point (to be placed) that are close to each other that are expecting the component's pair currently being assigned to the nozzle's pair. It is quite difficult to minimise the nozzle changes if we start by minimising the placement operation.

8.5.4 Discussion

We have proposed a methodology for an on-line scheduler that can construct an on-line schedule for the subsequent PCB points while the machine is performing pickup and placement operations. Results indicate that our proposed on-line scheduling algorithm with a greedy search heuristic is suitable for solving the component pickup-and-placement problem on hybrid pick-and-place machines and might be capable of producing good quality schedules. The on-line scheduling approach might continually produce a good schedule without incurring any 'significant cost' even if there is a change in the resources after the schedule is generated (allowing the machine to run continually). As a result, applying this approach to optimise the component pick-and-place operations on the hybrid pick-and-place machine may increase production throughput. Moreover, this might be achieved without paying an 'extra cost'. We also found the order of significant factors to be considered for generating a good quality schedule for the hybrid P&P machine is as follows, starting with the most significant:

a) minimise the nozzle changes;
b) maximise the simultaneous pickup;
c) maximise the simultaneous vision and
d) maximise the same feeder bank pickup (pickup both components from the same feeder bank).

Unfortunately, the use of database in this approach is very time consuming since the scheduler frequently communicates with a database to acquire the latest data. Therefore, we should reduce the number of communication between the scheduler and a database in order to generate a quick schedule. Of course, this will delay a respond to the spontaneous event. However, in the PCBA production line, the delay of responding to these spontaneous events in milliseconds or even a few seconds is not crucial, because it only causes a reduction in the machine throughput. Moreover, the spontaneous circumstances are not too frequently occurs. Therefore such delay can be ignored.

The subsequent work (section 8.6) extends these works by introducing a mechanical alignment procedure and considering component types which can have more than one packaging, and may require a different nozzle for picking and placing the same component type due to the different packaging types that are encountered. The latter procedure more closely mirrors the real-world, which previous works (section 8.5.2) have not addressed. Specifically, this further extends our previous contributions by introducing a novel weighted nozzle rank procedure. The heuristic can also be applied in on-line mode that we proposed in section 8.5.

To speed up the searching time, in the next section, we have transform the information from database into arrays such that the communication between the scheduler and database only happen before and after the schedule have been generated. This is a realistic strategy because the time taken by a scheduler to generate a complete schedule is less than 0.5 seconds (for this experiment) whilst the fastest sub tour of the HP-110 is about 1265 ms (i.e. a MA+SP sub tour). This means that any spontaneous events that occur during pick-and-place operation might be responded without stopping the machine operation.

8.6 Heuristic for Hybrid Pick-and-Place Machine

After some discussion with DIMA machine expert, we propose two nozzle selection heuristics, *OrderedNS* and *HybridNS*, with a component pick-and-place sequencing heuristic to optimise the HP-110. These works catered the issues listed in section 8.5.2 that have been overlooked in that work.

8.6.1 Nozzle Ranking Procedures

There are three nozzle ranking procedures, these being an ordered approach (R_0) and two weighted nozzle rank procedures, R_1 and R_2.

The R_0 approach ranks the nozzle pair starting with the maximum of *MAcounter,* then *SPcounter, Mcounter, MVcounter, SVcounter, SFcounter,* *STcounter* and *Scounter*. This ranking procedure is slightly difference from the one used in section 8.5.2 by including the *MAcounter, Mcounter, MVcounter* and *STcounter.* The ranking procedure is capable of producing a good quality schedule (i.e. good nozzle selection). However, the drawback is, this procedure may cause an unwanted nozzle changes due to a bad nozzle pair selection. That is, for example, we may choose a nozzle pair that has only one *MA* sub tour since the other nozzle pairs cannot be chosen because of *MAcounter=0* even though these nozzle pairs have many other sub tours. Therefore, in this case, we pay for at least two nozzle changes cost due to a decision of having an *MA* sub tour.

To overcome this problem, we propose a weighted nozzle rank procedure, R_1, to intelligently choose the best nozzle pair to be applied in each sub tour. The $R_1(z)$ function is computed based on the effectiveness of the sub tour's operation type and the appropriate counter's values of the nozzle pair. The larger the counter's values of the z^{th} nozzle pair, the more likely the nozzle pair is to be chosen. We use a weighted parameter, δ_s, to represent the effectiveness of the s^{th} sub tour's operation type. δ_s is calculated as follows:

$$\delta_s = \frac{E_s}{E_1} \tag{8.4}$$

where E_1 and E_s are the efficiency of the most efficient sub tour's operation type (i.e *MA+SP* sub tour) and the s^{th} sub tour's operation type, respectively. To obtain the E_s values, we compute the time taken for completing a sub tour and the machine throughput in components per hour (cph) for each pickup and placement operation type based on the average processing time of the HP-110 (see table 8.1). Table 8.10 shows the E_s values and summarises the machine throughput without a nozzle change operation based on one or two components pickup and placement operations. Table 8.10 shows the effectiveness of each pickup and placement operation type. It gives a significant clue as to how to devise a good strategy in constructing a good pickup and placement schedule. As shown in table 8.10, *MA+SP* is the most effective sub tour whilst *V* is the worst sub tour.

The δ_s values are shown in table 8.11. The weighted nozzle rank function of the z^{th} nozzle pair is:

$$R_1(z) = \sum_{s=1}^{S} \delta_s W_{zs} \qquad \textbf{(8.5)}$$

where W_{zs} is the s^{th} counter of the z^{th} nozzle pair and $S=14$. The relationship between *MAcounter, SPcounter* etc., δ_s, sub tour's operation type and the s^{th} is shown in table 8.11.

TABLE 8.10: THE THROUGHPUT OF THE HP-110 (A COMPLETE SUMMARY)

Pickup and placement operation type	**Time (ms)**	**E_s(cph)**
MA+SP	1265	5691
MA+SF	1665	4324
MV+SP	1680	4285
SV + SP	1680	4285
MA+DF	1725	4173
MA+SC	1875	3840
M	980	3673
SP	2080	3461
SV + SF	2080	3461
MV+SF	2080	3461
SV+DF	2140	3364
MV+DF	2140	3364
SV+SC	2290	3144
SF	2480	2903
DF	2540	2834
SC	2690	2676
V	1395	2580

To simplify the logic problem, reduce the computational time and avoid the use of too many variables, we only use an approximation to compute the W_{zs}. For example, W_{zs} of *MA+SP* sub tour is computed by taking the minimum of (*MAcounter, SPcounter*), which may not correct if all the mechanical alignment components for the nozzle pair cannot allow for simultaneous pickup and the value of the *SPcounter* is only contributed by the camera vision components. In this example we assume that if *MAcounter>0* and *SPcounter>0*, then we can have *MA+SP* sub tour.

TABLE 8.11: THE COUNTER'S RELATIONSHIP

s	**Sub tour**	**Counter**	δ_s
1	*MA+SP*	Min(*MAcounter, SPcounter)*	1.000
2	*MA+SF*	Min(*MAcounter, SFcounter)*	0.760
3	*MV+SP*	Min(*MVcounter, SPcounter)*	0.753
4	*SV + SP*	Min(SV*counter, SPcounter)*	0.753
5	*MA+DF*	Min(*MAcounter, STcounter –SFcounter)*	0.733
6	*M*	*Mcounter*	0.645
7	*SP*	*SPcounter*	0.608
8	*SV + SF*	Min(SV*counter, SFcounter)*	0.608
9	*MV+SF*	Min(MV*counter, SFcounter)*	0.608
10	*SV+DF*	Min(*SVcounter, STcounter –SFcounter)*	0.591
11	*MV+DF*	Min(M*Vcounter, STcounter –SFcounter)*	0.591
12	*SF*	*SFcounter*	0.510
13	*DF*	*STcounter–SFcounter*	0.498
14	*V*	*Scounter*	0.453

To increase the gaps among δ_s, we introduce another weighted nozzle rank function, R_2, as follows:

$$R_2(z) = \sum_{s=1}^{S} (\delta_s)^2 W_{zs} \tag{8.6}$$

In equation (8.6), δ_s is squared in order to increase the chances of selecting a nozzle pair that has many good quality sub tours. Without ignoring the lower efficient sub tours, R_2 heuristic is more likely to choose a nozzle pair that has many good quality sub tours compared to the R_1 heuristic. However, the R_2 heuristic does not just choose a nozzle pair that only has the most effective sub tour as in R_0 nozzle rank procedure. Therefore, the heuristic that uses the R_2 *nozzle ranking procedure might be* more capable of producing a good quality schedule compared to the heuristic that uses the R_0 *nozzle ranking procedure* and $R_1(z)$ function. However, in practice, the efficiency of each method is really dependent on the problem itself.

8.6.2 Nozzle Selection Heuristics

We develop two nozzle selection heuristics. These are an *OrderedNS* and a *HybridNS* nozzle selection heuristics. The *OrderedNS* heuristic is preliminary work that introduces a mechanical alignment procedure and considering component types that can have more than one packaging. It also includes the case of multi-pickup from the same component feeder in a sub tour. We then extend that work by introducing a *HybridNS* nozzle selection heuristic.

Since the works in section 8.5 did not consider the issues listed in section 8.5.2.1, dataset A also did not include the component packaging specification and the mechanical alignment components. Therefore, in this section, we need to use another dataset. We use two datasets in this work, these being Dataset B and Dataset DIMA. Dataset B (see appendix B) contains 30 PCB points, 10 component types, 14 component packages, 9 nozzles in the tool bank and 2 feeder banks. Dataset DIMA (see appendix C) is simulated data, provided by Dima SMT Systems, which has 156 PCB points, 26 component types, 26 component packages, 9 nozzles in the tool bank and 2 feeder banks.

8.6.3 An Ordered Nozzle Selection Heuristic

The ordered nozzle selection heuristic, *OrderedNS*, which is a constructive heuristic, is shown in figure 8.5. The algorithm begins by sorting the PCB points as in section 6.3.1.

Next, we create a list of decreasing ordered nozzle pairs, Z using an R_0 nozzle ranking procedure (the other two nozzle ranking procedures can also be applied but for this section, we only use the R_0 nozzle ranking procedure). The first top nozzle pair, $z=0$ where $z \in Z$ is chosen for the next pickups and placements. If there exist component(s) that need to be scheduled (*Q>0*), then the algorithm tries to schedule pairs of components that are possible for both simultaneous pickups and mechanical alignment (*MA+SP*) by the current nozzle's pair (i.e. if *SPcounter >0* and *MAcounter >0*). After scheduling all the *MA+SP* sub tours, then we try to schedule pairs of components that allow both mechanical alignment and same feeder pickup in a sub tour (*MA+SF*) without changing the nozzle pair until no more MA+SF sub tours can be scheduled. Similarly, we continue to schedule *SV+SP*, followed by *MV+SP, MA+DF, M, SP, SV+SF, MV+SF, SV+DF, MV+DF* and finally *SF* sub tours.

```
1. Sort the PCB points as in section 6.3.1.
2. Create nozzle pair list, Z and choose $z=0$ where $z \in Z$.
3. If $Q>0$;
  REPEAT
    3.1 Using the same nozzle pair:
      3.1.1 If possible, start schedule for MA+SP sub tours by choosing
            pairs of $p_{L0} \in P$ and $p_{R0} \in P$, and then $c_{pL0} \in C$ and $c_{pR0} \in C$.
            Reduce MAcounter($z_0$), SPcounter($z_0$), SFcounter, STcounter,
            Mcounter and Q accordingly. Similarly, schedule for
            MA+SF, SV+SP, MV+SP, MA+DF, M, SP, SV+SF, MV+SF,
            SV+DF, MV+DF and SF sub tours.
      3.1.2 If none of the nozzle pair can perform Step 3.1.1, then
            schedule for DF sub tours by choosing pairs of $p_{L0} \in P$ and
            $p_{R0} \in P$, and then $c_{pL0} \in C$ and $c_{pR0} \in C$. Reduce DFcounter($z_0$)
            and STcounter accordingly. Similarly, schedule for SC and
            reduce SFcounter, STcounter and Q accordingly.
      3.1.3 If none of the nozzle pair can perform Step 3.1.1 and 3.1.2,
            then schedule for V sub tours by choosing $p_{L0} \in P$ or $p_{R0} \in P$,
            and then $c_{pL0} \in C$ or $c_{pR0} \in C$. Reduce Scounter($z_0$) and Q
            accordingly.
    3.2 Nozzle changing:
        If $Q>0$ then re-generate the Z list and choose the best nozzle
        pair, $z=0$ where $z \in Z$.
  UNTIL $Q=0$.
4. Merge the single component sub tours.
5. Re-optimise the nozzle changes.
6. Avoid same component feeder pickup in a sub tour.
```

Figure 8.5: An *OrderedNS* heuristic algorithm.

Next, based on the PCB points that are left to be scheduled and the availabilities of component packages on the feeders, the *Z* list is regenerated. Again, the first top nozzle pair is chosen and the above processes are repeated (i.e. repeat step 3.1.1 in figure 8.5). However, at this stage, we try to re-apply the used nozzle (or tool) pair as much as possible. To do this, we rank the used nozzle pairs based on the number of sub tours that have been scheduled using the nozzle pair. Next, we search for the best nozzle pair (from the used nozzle pair list) to be applied. The obtained nozzle pair is compared to the first top nozzle pair, $z=0$. The best nozzle will be selected.

If none of the nozzle pair can perform step 3.1.1 in figure 8.5, then we try to schedule sub tours for different feeder pickups (*DF* sub tours) until no more *DF* sub tours can be scheduled, then similarly, we then schedule the *SC* sub tours. When none of the nozzle pair can perform step 3.1.1 and 3.1.2 in figure 8.5, we then try to schedule sub tours for single pickups (*V* sub tours) until no more *V* sub tours can be scheduled. The sequence of sub tours is determined based on the importance of the sub tour increasing the machine throughput (refer to table 8.10). For each sub tour, the appropriate PCB points are selected once the pickup components are scheduled. The appropriate counters of the nozzle pair are decreased accordingly after the PCB points are scheduled.

After scheduling all the available PCB points, we then proceed to Step 4 in figure 8.5 that will merge the single component sub tours.

In order to minimise the number of nozzle changes (step 5 in figure 8.5), we rearrange the sub tours such that the nozzle changes only happen whenever necessary. We begin by sorting the sub tours such that the sub tours, which use the same nozzle pair are consecutively indexed. If we must do a nozzle change, then we try to change only one nozzle if possible. This procedure also eliminates a reverse nozzle pair (i.e. $L_g= R_h$ and $R_g= L_h$ where $g \neq h$, $g<h$, $\{L_g,R_g,L_h,R_h\} \in T$) by swapping the left and right nozzles of the later nozzle pair and the appropriate PCB points and component packages (i.e. the h^{th} nozzle pair is converted to g^{th} nozzle pair). If one of the nozzles of the new pair is already used in the previous sub tour, but in a different side (i.e. $L_g= R_h$ or $R_g= L_h$ where $g \neq h$, $g<h$, $\{L_g,R_g,L_h,R_h\} \in T$), then we swap the left and right nozzles of the new

nozzle pair and the appropriate PCB points and component packages (such that $L_g = L_h$ or $R_g = R_h$ whichever applicable). If necessary, we try to swap the nozzles and components such that we can increase the number of sub tours that are able to utilise the most used nozzle pair and eliminate the least used nozzle pair.

Finally, to further improve the schedule, if possible, we swap a component package in a same component feeder (*SC*) sub tour with a component package in the other sub tour in order to avoid same component feeder pickups (step 6 in figure 8.5). However, we avoid swapping *SC* (*SV+SC* etc.) with *SP* (*SV+SP* etc.) since we do not want to sacrifice the *SP* sub tours. We also avoid swapping a mechanical aligned component with a vision component.

8.6.4 An Ordered Nozzle Selection Heuristic: Testing and Results

The work in this section only uses dataset B (see appendix B) because dataset DIMA (see appendix C) was not available when the research was conducted.

Table 8.12 shows an example of the result obtained by the *OrderedNS* before applying Step 4, 5 and 6 in figure 8.5, whilst table 8.13 shows an improved schedule. The schedule in table 8.12 caused 4 nozzle changes with *CT*= 36060ms (for assembling 30 components) and producing 2995cph whilst an improved schedule (result in table 8.13) also caused 4 nozzle changes with *CT*=35320ms and producing 3057cph. Results show that the machine's throughput is improved from 2995cph to 3057cph (2.07% improvement).

The Schedule in table 8.12 has four single sub tours, these being the 8th, 12th, 13th and 16th sub tours, which will be merged by the merging procedure. The nozzles, packages and PCB points in sub tour 14 and 15 (table 8.12) are swapped left-right by the re-optimise nozzle change procedure. Finally to avoid pickups from the same component feeder, the component in the 9th sub tour (table 8.12) is swapped with the other sub tour.

TABLE 8.12: THE RESULT OBTAINED (WITHOUT STEP 4, 5 AND 6 IN FIGURE 8.5)

Sub tour	**Left nozzle**	**Right nozzle**	**Left PCB ID**	**Right PCB ID**	**Left package**	**Right package**	**Operation type**	**$\eta(j)$**
0	2	4	1	18	C	A	MA+SF	0
1	2	4	8	12	C	A	MA+SF	0
2	2	4	14	19	C	A	MA+SF	0
3	2	4	20	22	C	A	MA+SF	0
4	2	4	4	3	G	D	SV+SF	0
5	2	4	6	2	G	D	SV+SF	0
6	2	4	30	24	G	D	SV+SF	0
7	2	4	21	13	G	J	SV+DF	0
8	**2**	**4**	**NONE**	**23**	**NONE**	**J**	**V**	**0**
9	**8**	**16**	**5**	**9**	**E**	**E**	**MA+SC**	**2**
10	8	16	25	16	M	O	SV+SP	0
11	8	16	15	29	M	O	SV+SP	0
12	**8**	**16**	**27**	**NONE**	**E**	**NONE**	**M**	**0**
13	**8**	**16**	**28**	**NONE**	**M**	**NONE**	**M**	**0**
14	1	64	10	17	B	L	MA+DF	2
15	1	64	7	26	B	L	MA+DF	0
16	**1**	**64**	**11**	**NONE**	**B**	**NONE**	**M**	**0**

TABLE 8.13: THE FINAL SCHEDULE GENERATED BY AN ORDEREDNS HEURISTIC

Sub tour	Left nozzle	Right nozzle	Left PCB ID	Right PCB ID	Left package	Right package	Operation type	η(j)
0	2	4	1	18	C	A	MA+SF	0
1	2	4	8	12	C	A	MA+SF	0
2	2	4	14	19	C	A	MA+SF	0
3	2	4	20	22	C	A	MA+SF	0
4	2	4	4	3	G	D	SV+SF	0
5	2	4	6	2	G	D	SV+SF	0
6	2	4	30	24	G	D	SV+SF	0
7	2	4	21	13	G	J	SV+DF	0
8	8	4	27	23	E	J	MV+SF	1
9	8	16	5	9	E	E	MA+SC	1
10	8	16	25	16	M	O	SV+SP	0
11	8	16	15	29	M	O	SV+SP	0
12	8	1	28	11	M	B	MA+DF	1
13	64	1	17	10	L	B	MA+DF	1
14	64	1	26	7	L	B	MA+DF	0

Based on our experiment on a Pentium 4, 1.5Ghz, 256 MB RAM computer, we obtained a complete schedule in about 0.3 seconds. Therefore the machine may start the pickup and placement operation at 0.3 seconds after the PCB data is downloaded into the machine or, perhaps, immediately after the board (i.e. PCB) is ready (clamped on the PCB table) if the board loading time is more than 0.3 seconds (for example).

8.6.5 A Hybrid Nozzle Selection Heuristic

A hybrid nozzle selection with a component pickup-and-placement sequencing heuristic (*HybridNS*), which is a constructive heuristic, is shown in figure 8.6 (mostly adopted from section 8.6.3). This heuristic is an enhancement of an *OrderedNS* heuristic. Steps 1 to 6 are the same as an *OrderedNS* heuristic (see section 8.6.3). However, in this approach, we investigate the effectiveness of the three nozzle ranking procedures (see section 8.6.1) by utilising all the three nozzle ranking procedures to create the *Z* list (one at each test).

We then re-optimise the nozzle changes (step 7 in figure 8.6) as in step 5 in figure 8.6. This has to be performed since the previous step (step 6 in figure 8.6) might create a new nozzle pair. Step 8 in figure 8.6 will ensure that all used nozzle pairs are feasible. Not all the nozzles have a duplicate copy in the tool bank (i.e. most of the nozzles are unique). Therefore, if we have an identical left-right nozzle pair whilst the tool bank only has one copy of this nozzle, then the nozzle pair is infeasible. Infeasible nozzle pairs might be created by steps 4, 5, 6 and 7 in figure 8.6).

Finally, we apply a simple re-optimise nozzle pair (step 9 in figure 2) that is guaranteed not to produce an infeasible nozzle pair.

8.6.6 A Hybrid Nozzle Selection Heuristic: Testing and Results

As this is an unexplored problem, we can only compare among our approaches to demonstrate the performance of the approaches. In this section we use dataset B (see Appendix B) and dataset DIMA (see Appendix C).

```
1. Sort the PCB points as in section 6.3.1.
2. Create a nozzle pair list, Z and choose z=0 where $z \in Z$.
3. If Q>0;
  REPEAT
    3.1  Using the same nozzle pair:
      3.1.1 If possible, start schedule for MA+SP sub tours by choosing
            pairs of $p_{L0} \in P$ and $p_{R0} \in P$, and then $c_{pL0} \in C$ and $c_{pR0} \in C$.
            Reduce MAcounter($z_0$), SPcounter($z_0$), SFcounter, STcounter,
            Mcounter and Q accordingly. Similarly, schedule for MA+SF,
            SV+SP, MV+SP, MA+DF, M, SP, SV+SF, MV+SF, SV+DF,
            MV+DF and SF sub tours.
      3.1.2 If none of the nozzle pair can perform Step 3.1.1, then schedule
            for DF sub tours by choosing pairs of $p_{L0} \in P$ and $p_{R0} \in P$, and
            then $c_{pL0} \in C$ and $c_{pR0} \in C$. Reduce DFcounter($z_0$) and STcounter
            accordingly. Similarly, schedule for SC and reduce SFcounter,
            STcounter and Q accordingly.
      3.1.3 If none of the nozzle pair can perform Step 3.1.1 and 3.1.2,
            then schedule for V sub tours by choosing $p_{L0} \in P$ or $p_{R0} \in P$, and
            then $c_{pL0} \in C$ or $c_{pR0} \in C$. Reduce Scounter($z_0$) and Q
            accordingly.
    3.2 Nozzle changing:
        If Q>0 then re-generate the Z list and choose the best nozzle pair,
        z=0 where $z \in Z$.
  UNTIL Q=0.
4. Merge the single component sub tours.
5. Re-optimise the nozzle changes.
6. Avoid same component feeder pickup in a sub tour.
7. Re-optimise the nozzle changes (same as step 5).
8. Eliminate the infeasible sub tour.
9. Simple re-optimise the nozzle changes.
```

Figure 8.6: A *HybridNS* with a component pickup-and-placement sequencing heuristic.

For each dataset, we perform 200 runs. Since there is no random element in our heuristic, any run will obtain the same result for the same dataset. Therefore, in this experiment, we need to modify the contents of these datasets to demonstrate their effectiveness. For each run, we randomly modify the specification of the component recognition and the nozzle assignment of each component package. At each run, at most four nozzles are randomly selected to be assigned to a component package. Therefore, each run is effectively a different problem instance. However, for each run, the same problem instance is used to test the performance of the three nozzle ranking approaches, these being an R_0 and the two weighted nozzle rank heuristics, R_1 and R_2. Based on the cycle time (*CT*) of the schedule obtained by each approach, we compute the component per hour (cph) to measure the machine throughput of each solution (denoted as M_0, M_1 and M_2 for components per hour for R_0, R_1 and R_2, respectively).

The relative change in M_1 over M_0, denoted as I_1 ($I_1 = (M_1-M_0)*100/M_0$), and the relative change in M_2 over M_0, denoted as I_2 ($I_2 = (M_2-M_0)*100/M_0$), are shown in figure 8.7 (dataset B) and figure 8.8 (dataset DIMA). Both figures show that in some cases, the R_1 and R_2 are better than the R_0 (i.e. positive improvement, which are the points above *X*-axis), whilst in certain cases the R_0 outperformed the other two ranking procedures (i.e. negative improvement, which are the points below *X*-axis).

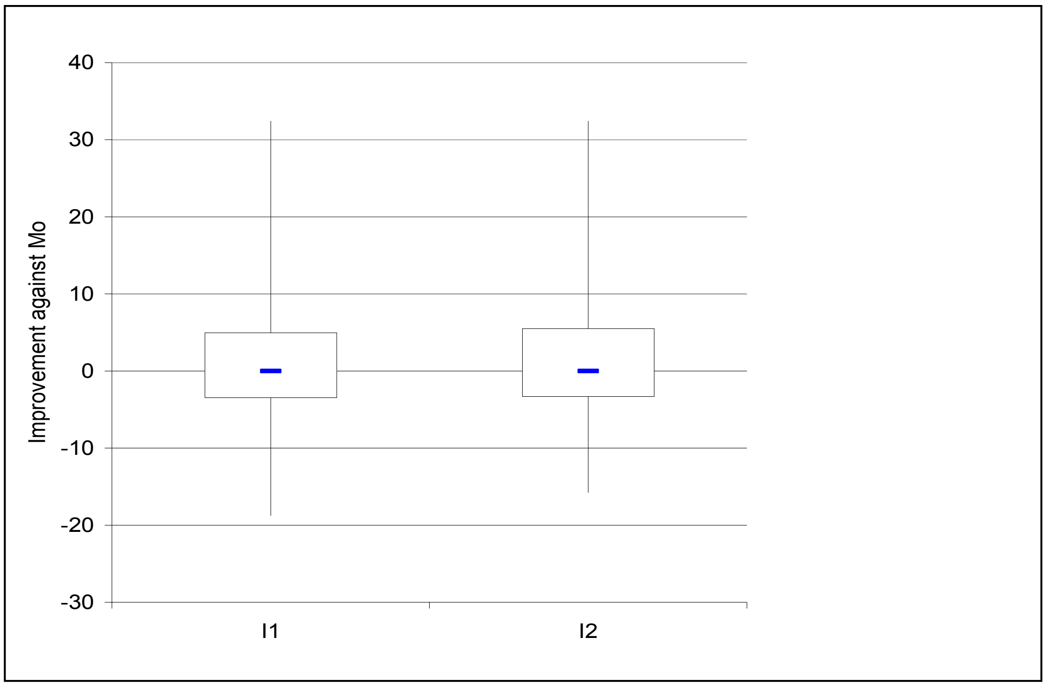


Figure 8.7: I_1 and I_2 for dataset B.

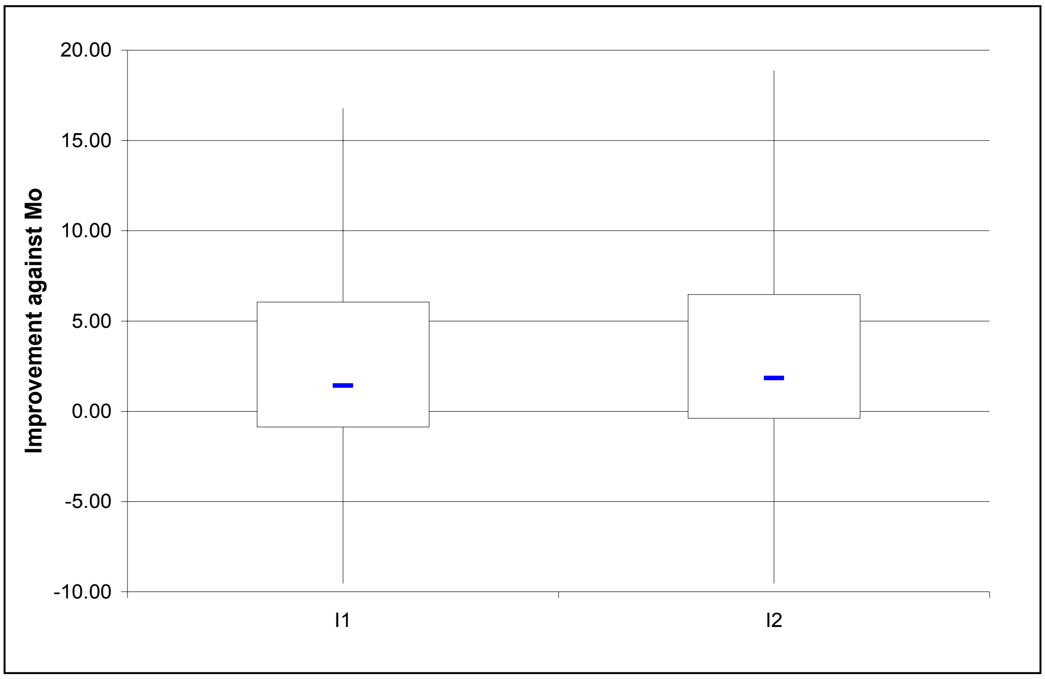


Figure 8.8: I_1 and I_2 for dataset DIMA.

Table 8.14 summaries the result of 200 runs. It shows that, for dataset B, 48% and 49% of the 200 runs reported improved solution quality when comparing M_1 and M_2 against M_0, respectively. For dataset DIMA, we obtained improved solution quality of 65% and 70% of the 200 runs when comparing M_1 and M_2 against M_0, respectively. Figure 8.9 shows the distribution of the result when comparing M_1 and M_2 against M_0 for dataset DIMA.

Based on an ANOVA (analysis of variance) test on I_1 and I_2 (dataset B and DIMA), there is no significant difference in the performance of R_1 and R_2, when compared to the R_0 (F-ratio=0.04 and 0.42, respectively for α=0.01). We can graphically observe these results by presenting a frequency distribution graph (see figure 8.9).

TABLE 8.14A: THE SUMMARY OF 200 RUNS COMPARING M_1 AND M_2 AGAINST M_0 (DATASET B)

	Minimum (%)	**Average (%)**	**Maximum (%)**	**Improvement > 0 (%)**	**Improvement = 0 (%)**	**Improvement < 0 (%)**
I_1	-18.77	1.39	32.43	48.00	8.50	43.50
I_2	-15.78	1.54	32.43	49.00	8.00	43.00

TABLE 8.14B: THE SUMMARY OF 200 RUNS COMPARING M_1 AND M_2 AGAINST M_0 (DATASET DIMA)

	Minimum (%)	**Average (%)**	**Maximum (%)**	**Improvement > 0 (%)**	**Improvement = 0 (%)**	**Improvement < 0 (%)**
I_1	-9.54	2.43	16.79	65	2.5	32.5
I_2	-9.54	2.77	18.88	70	1.5	28.5

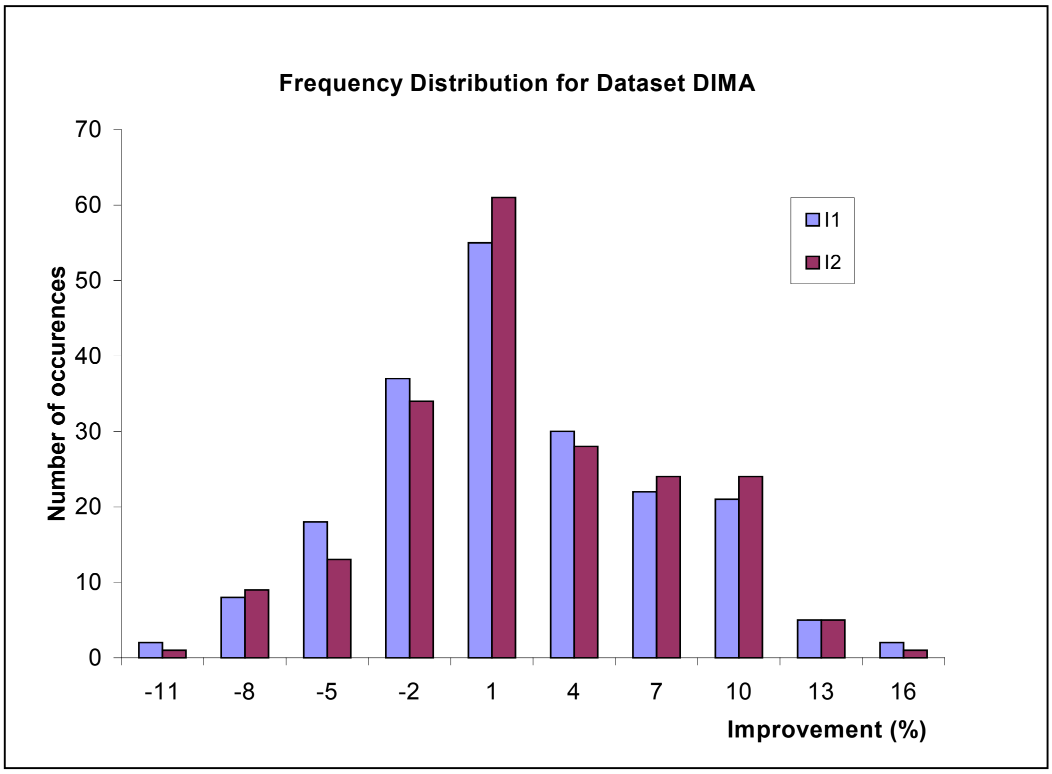


Figure 8.9: A frequency distribution of the I_1 and I_2 for dataset DIMA.

Table 8.14 also shows that, on average R_2 slightly outperformed R_0 in about 1.54% and 2.77% for dataset B and dataset DIMA, respectively. Whilst, R_1 is slightly superior to the R_0 in about 1.39% and 2.43% for dataset B and dataset DIMA, respectively. From this experiment, the best improvement obtained by the R_1 against the R_0 are 32.43% for dataset B and 16.79% for dataset DIMA, whilst the R_2 gains 32.43% for dataset B and 18.88% for dataset DIMA over the R_0. On the contrary, the worst result obtained by the R_1 against the R_0 are -18.77% for dataset B and -9.54% for dataset DIMA, whilst the R_2 obtained -15.78% for dataset B and -9.54% for dataset DIMA over the R_0. This indicates that each of the approaches have their own strengths and weaknesses. For example, the R_0 nozzle ranking approach may perform best when the dataset has many mechanical aligned components, that may incur many *MA+SP, MA+SF, MA+DF* or *MA+SC* sub tours, which are among the few best sub tours. The R_0 nozzle ranking approach also chooses those sub tours compared to the other sub tours. Therefore, as a result the R_0 nozzle ranking approach might be capable of producing a schedule, which has many good quality sub tours compared to the R_1 and R_2 nozzle ranking approaches. However, due to a decision of first searching for a nozzle pair that has *MA+SP* sub tours, then *MA+SF* sub tours etc., R_0 nozzle ranking approach only considers the counters of the highest quality sub tour for each nozzle pair, without considering the value of the counters of the lower quality sub tours. This might lead to selecting a bad nozzle pair that may cause unnecessary nozzle changes. When this case happen, the R_0 nozzle ranking approach might produce a bad quality schedule even though the generated schedule contains many good quality sub tours since there are unnecessary nozzle changes, which drastically reduced the machine's throughput since tool changes are very time consuming. This explains why sometimes R_2 and R_1 gain 32.43% over the R_0 and sometimes the R_0 heuristic is better than the other two heuristics. There is a tradeoff between having many good quality sub tours and minimising the tool change operations. Therefore, the R_1 and R_2 heuristics are introduced to overcome the problem. Compared to R_1, the R_2 heuristic places more emphasis on selecting a tool pair, which has many good quality sub tours without ignoring the existence of the lower quality sub tours. As a result, on average, the R_2 heuristic is capable of producing a better quality schedule compared to the R_1 and the R_0 nozzle ranking heuristics.

Based on our experiments on a Pentium 4, 1.5Ghz, 256 MB RAM computer, we obtained a complete schedule in about 0.3 seconds (for Dataset B, which has N=30 and K=10) after the PCB data was downloaded into the machine. To investigate the scalability of the result, we carried out another experiment that varied the value of N whilst the other parameter values (i.e. K, board size, feeder setup, component packaging assignment etc.) were fixed. The following N values were chosen; 30, 60, 90, 120, 240, 480, 960, 1920, 2880, 3840, 5760, 7680 and 10000. The maximum value of N=10000 was chosen due to the fact that, currently, it is very rare to have more than 10000 components on a PCB. Figure 8.10 shows the result of this experiment. We can observe from figure 8.10 that, for all cases, the computation time increases with an increase in *N*. Figure 8.10 shows that for a small value of *N* (i.e. *N*<3000), the computation time linearly increases with an increase in *N* (for all heuristics). However, for larger values of *N* (i.e. *N*>3000), it appears exponential but still appears manageable, as the value of N is never likely to be very large due to the problem domain.

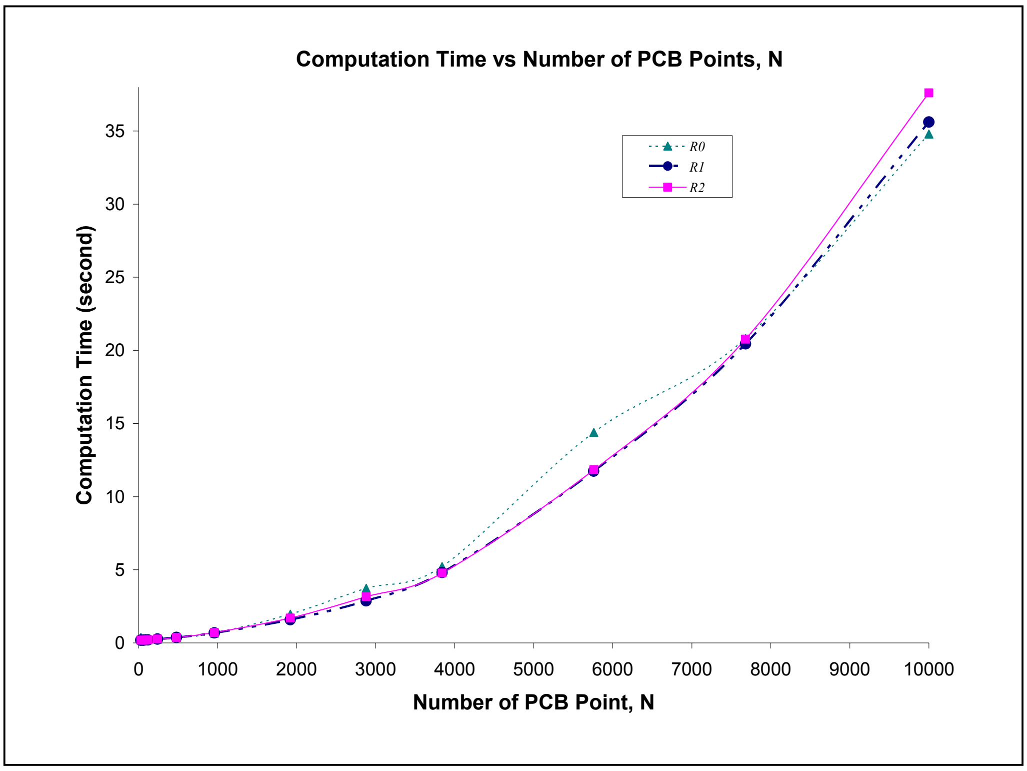


Figure 8.10: The computation time of the R_0, R_1 and R_2 nozzle ranking approaches which varies *N* parameter value.

8.6.7 Discussion

We have proposed a methodology for a constructive heuristic to schedule a component pick-and-place operation for a new hybrid pick-and-place machine. The aim is to minimise the assembly cycle time. The proposed heuristic intelligently chooses a nozzle pair based on a new weighted nozzle rank procedures or an R_0 procedure. The nozzle is ranked based on the effectiveness of the sub tour's operation type and the appropriate counter values of the nozzle pair. The larger the counter values of the nozzle pair, the more likely the nozzle pair is to be chosen. By using a weighted nozzle rank procedure, we overcome the tradeoff issues of having many good quality sub tours or minimising the tool changes operations. We have addressed the importance of choosing a proper nozzle group in maximising the machine throughput since a nozzle change operation is time consuming. As this is an unexplored problem, we can only compare among our approaches and datasets. On average, we found that a weighted nozzle rank approach was slightly superior to the ordered (R_0) approach, although this difference was not statistically significant. However, theses nozzle ranking approaches have their own strength and weaknesses. Which approach is better than the other is dependent on the problem instance that is being solved.

8.7 Summary

In this chapter, we have addressed a real-world scheduling problem arising in a PCBA production line, in particular, an optimisation problem of the hybrid pick-and-place machine. Most of the issues such as different component packaging for the same component type, component feeder transportation time and component specific nozzles are usually ignored or overlooked by many researchers.

A framework for an on-line scheduling approach that utilises a database and an interrupt feature has been proposed. The proposed on-line scheduler could continuously schedule the subsequent PCB points while the machine is performing pickup and placement operations. Without incurring 'any significant cost', the scheduler might continuously repair the schedule if any spontaneous events occur while the machine is running. As a result, applying the proposed framework for on-line scheduling might increase the machine throughput. Unfortunately, the use of a database in this approach is very time consuming since the scheduler frequently communicates with the database to acquire the latest data. To speed up the search time, we transformed the information from a database into arrays such that the communication between the scheduler and database only happen before and after the scheduler have been generated. This is more realistic strategy because the time taken by our scheduler to generate a complete schedule is less than 0.5 seconds (for example) whilst the fastest sub tour of the HP-110 machine is about 1265 milliseconds (i.e. *MA+SP* sub tour). This means that any spontaneous events that occur during pick-and-place operation might be addressed without stopping the operation of the machine.

Three nozzle rank procedures have been presented. These are an R_0 and the two weighted nozzle rank procedures, R_1 and R_2. The nozzle is ranked based on the effectiveness of the sub tour's operation type and the appropriate counter values of the nozzle pair. The larger the counter values of the nozzle pair, the more likely the nozzle pair is to be chosen. However, it is a tradeoff issue of having many good quality sub tours or minimising the tool changes operation. By using a weighted nozzle rank procedures, we overcome the tradeoff issue. As this is an unexplored problem, we can only compare among our approaches and datasets. On average, we found that a weighted nozzle rank approach was slightly superior to the ordered (R_0) approach, although this difference was not statistically significant.

We have addressed the importance of choosing a proper nozzle group in maximising the machine throughput since a nozzle change operation is time consuming. Hence, we proposed two nozzle selection heuristics that are an *OrderedNS* and a *HybridNS*. These heuristics give highest priority to minimising the number of nozzle changes in sequencing the pick-and-place operations.

We also found the order of significant factors to be considered for generating a good quality schedule for the hybrid pick-and-place machine is as follows, starting with the most significant:

a) minimise the nozzle changes;

b) maximise the multi-pickup of mechanical aligned component (MA sub tour);

c) maximise the simultaneous pickup;

d) maximise the simultaneous vision pickup and

e) maximise the same feeder bank pickup (pickup both components from the same feeder bank).

The work reported in this chapter was conducted based on a simulation dataset given by DIMA's machine expert and our randomly generated datasets (based on the problem description by DIMA's machine expert). In order to test the proposed approach on the real-world machine, some modifications might be required to ensure correct communication between the scheduler and other software on the SMD placement machine. The proposed approach might be applicable to other SMD placement machines which have similar characteristics.

The proposed approaches in this chapter were only focused on a constructive heuristic that is machine specific. However, a general solution framework might be applicable in solving other machine types.

The next chapter concludes all our studies in optimising the SMD placement machine. The chapter also discusses some suggested directions for future work.

Chapter 9

Conclusions and Future Work

9.1 Introduction

This chapter summarises the research reported in this thesis. Section 9.2 highlights the major conclusions of the research whilst the contributions are discussed in section 9.3. Finally, section 9.4 suggests some possible future research directions.

9.2 Research Work Summary

The research conducted in this thesis was motivated by the demand of optimising the SMD (surface mount device) placement machine where so far, the PCB (printed circuit board) machine vendors and software companies are still not capable of solving even a single machine optimisation problem efficiently. Moreover, many reported works in this area are too abstract, which might not be efficient for solving real-world machine problems. Therefore, this research has studied a real-world machine problem and proposed a heuristic to optimise the machine that tried to satisfy many major optimisation factors of the machine.

A comprehensive survey (chapter 3) on single SMD placement machine optimisation was carried out. Since most of the heuristics developed for solving the single SMD placement machine optimisation problem are machine specific, this work was carried out to identify the relationships between models, assembly machine technologies and heuristic methods. The survey proposed five categories of SMD placement machines based on their specifications and operational methods; these being dual delivery, multi-station, turret-type, multi-head and sequential pick-and-place SMD placement machines. These grouping aimed to guide future researchers in this field to have a better understanding of the various machine specifications and operational methods, and subsequently enable them to apply or even design heuristics, which are more appropriate to the machine characteristics and the operational methods. The survey revealed some of the optimisation issues in each of the SMD placement machine category. In addition, the survey also classified the single SMD placement machine optimisation problem into five sub problems and highlighted the optimisation issues in each category. These being a feeder setup, placement sequencing, nozzle optimisation, component retrieval plan and motion control sub problems.

A revised dynamic pick-and-place point (DPP) specification approach called Chebychev DPP (CDPP) had been proposed in chapter 4. The CDPP motion control eliminated the unnecessary movement by looking forward to the next PCB coordinate when determining the current pickup location and looking forward the next feeder slot when determining the current placement location. The CDPP formulations are constructed based on the aims of minimising robot assembly time, feeder movements and PCB table movements. The main difference between the CDPP model and the previous DPP (and EDPP) was that the CDPP calculated the robot arm movement distance as the maximum of the movement in Y or the movement in X (a chebychev distance) since the robot arm can move in X-axis and Y-axis concurrently, whilst the previous DPP (and EDPP) calculated the robot arm movement as a euclidean distance. This work has shown an improvement compared to Wang’s DPP approach. Therefore, the work was extended by integrating the CDPP approach with a novel a triple objective function to improve the feeder setup in order to gain even better results. The triple objective function aims to minimise the CT (assembly cycle time) together with minimising the feeder carrier and PCB table movements. In summary, minimising the triple objective function strategy might provide better solutions (for feeder setup) compared to the strategies of minimising the CT and minimising the exchange frequency.

As the CPU (central processing unit) of the SMD placement machine might always in the idle state while the robot arm is moving, picking and placing components and the robot arm is normally an interrupt driven I/O (input/output) device, we can make use of the CPU free time to improve the initial schedule. Hence, chapter 5 introduced a theoretical on-line scheduling approach that might employ the CPU free time to improve the initial schedule. A greedy constructive heuristic was employed to generate an initial solution then a random descent method was used to improve the initial schedule. Results shows that the CT improved by 36.60% (dataset 1) and 43.29% (dataset 2) over the initial schedule.

Various heuristics have been applied to optimise the component pick-and-place sequence of a theoretical multi-head SMD placement machine. These are greedy local searches, hyper-heuristics and variable neighbourhood search.

Since the component pick-and-place sequencing problem of a theoretical multi-head SMD placement machine is confronted with various optimisation factors such as optimisation of the pickup sequence, the placement sequence, pipette assignment, sub-tour grouping and sequencing the sub-tour, it causes difficulties in devising a good strategy to minimise the assembly cycle time. However, by applying a hyper-heuristic approach, we do not have to concern ourselves with the trade-off between the optimisation of the important factors as this will be catered for within the hyper-heuristic. The ability of hyper-heuristics approaches in solving the component pick-and-place sequencing problem of a theoretical multi-head SMD placement machine had been demonstrated in chapter 6. A greedy search hyper-heuristic that randomly calls low-level heuristics (LLHs) and accepts any returned solution has been presented. The greedy search hyper-heuristic operates with six LLHs. Instead of using a simple descent method in each local search (i.e. LLH), as used in chapter 5, the three neighbourhood search techniques to determine a move from one schedule to another schedule were employed. These being: random descent, random move and steepest descent approaches.

To further investigate the effectiveness of hyper-heuristic approach in solving the component pick-and-place sequencing problem, a Monte Carlo based hyper-heuristic was introduced (chapter 6). In the case study, the Monte Carlo hyper-heuristic randomly calls a LLH. However, one can intelligently choose the LLH to be applied. The new solution returned by the low-level heuristic will be accepted based on the Monte Carlo acceptance criterion. The Monte Carlo acceptance criterion always accepts an improved solution. Worse solutions will be accepted with a certain probability, which decreases with worse solutions, in order to escape local minima. Three hyper-heuristics based on a Monte Carlo method have been developed. These being a Linear Monte Carlo (LMC), an Exponential Monte Carlo (EMC) and an Exponential Monte Carlo with counter (EMCQ). Experimental results show that, in general, LMC and EMCQ show almost equal performance, but EMCQ has a better formulation, which includes the intensification (time, t) and diversification (counter for consecutive unimproved, Q) factors. Moreover, EMCQ does not appear to be parameter sensitive, unlike LMC whose performance does change when the parameters are adjusted. Therefore, it is not only a fast heuristics, but also robust as the parameters do not have to be tuned.

Since the pick-and-place sequencing problem of the theoretical multi-head SMD placement machine requires various neighbourhood structures, a Variable Neighbourhood Search might be a suitable choice. Hence, a Variable Neighbourhood Search for solving the component pick-and-place sequencing of multi-head placement machine has been proposed in chapter 7. A basic version of a VNS, that is VNS-1, VNS-2 and VNS-3 have been proposed. The VNS-1 is a descent heuristic that operates with random descent local search. VNS-2 is also a descent heuristic but operates with EMC (Exponential Monte Carlo) local search. The EMC local search always accepts an improved solution and probabilistically accepts a worse solution depending on the degree of worsening (δ). The VNS-3 is a descent-ascent heuristic with EMCQ (Exponential Monte Carlo with counter) acceptance criterion that operates with an EMC local search. Our experimental results show that the VNS-1, VNS-2 and VNS-3 do not perform well compared to some hyper-heuristics approaches presented in this work. Therefore, a Variable Neighbourhood Search with Exponential Monte

Carlo (VNMS) acceptance criterion has been proposed. The VNMS is a descent-ascent heuristic that operates on three sets of neighbourhood structures that are grouped together based on three different local search/operator approaches. Each group contains six neighbourhood structures. The first two sets use a steepest descent and EMC local search whilst the third set uses a random 3-opt operator. The solution returned by a local search, after exploring a neighbourhood structure, will be accepted based on the EMCQ acceptance criterion. The EMCQ acceptance criterion always accepts an improved solution. However, the probability of accepting a worse solution increases as δ decreases and the counter of consecutive none improvement iterations, Q, increases. VNMS begins by exploring the neighbourhood in the steepest descent group until no further improvement or some other termination criteria are met. Secondly, the EMC neighbourhood will be explored to divert the search direction by probabilistically accepting some worse solution. After exploring all neighbourhood structures in the EMC neighbourhood group and the search is trapped in a local optimum, then a shaking procedure is applied by randomly selecting a neighbour in random 3-opt neighbourhood structures using a 3-opt operator. The trial solution will be evaluated by the EMCQ acceptance criterion. If it is not accepted, another trial will be performed until the solution is accepted or another termination criterion is met. Once the trial solution produced by the shaking procedure is accepted, the procedures are repeated (until the termination criterion is met). The solution obtained from the shaking procedure will be used as an incumbent solution. Results show that the VNMS can produce good quality and stable results. Therefore, the VNMS might be more reliable compared to the hyper-heuristic approaches tested in this work.

An on-line scheduler that can construct an on-line schedule for the subsequent PCB points while the machine is performing pickup-and-placement operations was presented in chapter 8. This approach is different from the work presented in chapter 5, which only allowed the machine to begin a pickup-and-placement operation once a complete schedule was available, prepared an improved schedule for the subsequent PCB only (not for the current PCB being processed), ignored the nozzle change operation (it assumed that all components can be picked up by the same nozzle) and modelled different types of machine specification. This work is more closely mirrors the real-world, which previous works have not addressed. Results indicate that the theoretical on-line scheduling algorithm with a greedy search heuristic might be suitable for solving the component pick-and-place problem on hybrid pick-and-place SMD placement machines and might produce a good quality schedule. The on-line scheduling approach might continually produce a good schedule without incurring any cost even if there is a change in the resources after the schedule is generated (allowing the machine to run continually). As a result, applying on-line scheduling with a greedy search heuristic to optimise the components' pickup-and-placement operation on the hybrid pick-and-place SMD placement machine might increase production throughput. Moreover, this might be achieved without paying extra cost.

Three nozzle rank procedures have been presented (chapter 8). These are an *Ordered* and the two weighted nozzle rank procedures, *F(z)* and $F^2(z)$. The nozzle is ranked based on the effectiveness of the sub tour's operation type and the appropriate counter values of the nozzle pair. The larger the counter values of the nozzle pair, the more likely the nozzle pair is to be chosen. However, it is a tradeoff issue of having many good quality sub tours or minimising the tool changes operation. By using a weighted nozzle rank procedure, we might overcome the tradeoff issue. As this is an unexplored problem, we can only compare among our approaches and datasets. On average, we found that a weighted nozzle rank approach is superior to an *Ordered* approach.

In chapter 8, we also addressed the importance of choosing a proper nozzle group in maximising the machine throughput since a nozzle change operation is time consuming. Hence, we proposed two nozzle selection heuristics that are an *OrderedNS* and a *HybridNS*. These heuristics give highest priority to minimising the number of nozzle changes in sequencing the pick-and-place operations. We also found the order of significant factors to be considered for generating a good quality schedule for the hybrid pick-and-place machine is as follows, starting with the most significant:

a) minimise the nozzle changes;

b) maximise the multi-pickup of mechanical aligned component (MA sub tour);

c) maximise the simultaneous pickup;

d) maximise the simultaneous vision pickup and

e) maximise the same feeder bank pickup (pickup both components from the same feeder bank).

The most challenging part in this work was to understand the real-world machine problems and designing a good heuristic for optimising this machine by considering most of the important optimisation issues of this machine. Many optimisation issues, which were rarely (or have never been) addressed such as tray feeder optimisation, nozzle optimisation, simultaneous pickup, multiple pickups from the same component feeder, different component packaging of the same component type, which require different nozzle type etc. have been addressed in this work. Unfortunately, due to the complexity of the tray feeder problem, the optimisation of this problem is left for future research.

9.3 Contributions

The work carried out in this thesis has led to the following contributions:

a) A Monte Carlo based acceptance criteria has been designed, which has led to the introduction of a Monte Carlo based hyper-heuristic. This has further improved the optimisation of the multi- head SMD placement machine.

   The Monte Carlo acceptance criterion always accepts an improved solution. Worse solutions will be accepted with a certain probability in order to escape local minima. Three types of acceptance criteria based on a Monte Carlo based acceptance criteria have been introduced, these being Linear Monte Carlo (LMC), Exponential Monte Carlo (EMC) and Exponential Monte Carlo with counter (EMCQ).

   The LMC probability is computed by $(M-\delta)$ where M is a constant valued between 0 and 100 and $\delta=f(S_c)-f(S_0)$.

   The EMC probability is computed by $e^{-\delta}$ in which the probability of accepting a worse solution decreases as the δ increases.

The EMCQ probability is computed by $e^{-\theta/\tau}$ where $\theta=\delta*t$ and $\tau=\rho(Q)$. t is a computation time (in our case we use minutes as a unit time). θ and τ are defined such that we ensure that the probability of accepting a worse solution decreases as the time increases and δ increases. The factor of time is included in this formulation as an intensification factor. At the beginning of the search, the moderately worse solution is more likely to be accepted, but as the time increases the worse solution is unlikely to be accepted. However, the probability of accepting a worse solution increases as the counter of consecutive none improvement iterations, Q increases. This is a diversification factor. $\rho(Q)$ is a function to intelligently control the Q. In this work we use $\tau=v*Q$ where $0\leq v\leq 1$, in order to limit the acceptance probability. However, the preliminary experiment on parameter sensitivity of v shows that the EMCQ algorithm with $v=1$ performs the best. In fact, the EMCQ is not sensitive to the value of v (i.e. any value of v produces almost the same quality of result).

The Monte Carlo hyper-heuristic randomly calls a LLH (low-level heuristic). However, one can intelligently choose the LLH to be applied. The new solution returned by the LLH will be accepted based on the Monte Carlo acceptance criterion.

b) A new heuristic, which is a revised dynamic pick-and-place point (DPP) specification approach called Chebychev DPP (CDPP) and a triple objective function that attempts to minimise the assembly cycle time together with the minimisation of the movement of the feeder carrier and the PCB table, has been developed.

A CDPP approach tries to eliminate unnecessary movement by looking forward to the next PCB coordinate when determining the current pickup location and looking forward to the next feeder slot when determining the current placement location. Instead of only searching for a minimum assembly cycle time, the CDPP approach is looking for a minimum assembly cycle time and a reduction in feeder carrier and PCB table movement as far as possible. The triple objective function aims to minimise the CT whilst also minimising the feeder and PCB table movement.

However, the main objective remains to minimise the CT but it would be beneficial if we can also minimise the feeder and PCB table movement. Reducing these movements may prolong the life cycle of SMD placement machine even if it does not affect the throughput rate of the machine.

c) A Variable Neighbourhood Monte Carlo Search (VNMS) heuristic, which employs a variable neighbourhood search technique with an Exponential Monte Carlo acceptance criterion, has been developed.

   The VNMS is a descent-ascent heuristic that operates on three sets of neighbourhood structures that are grouped together based on three different local search/operator approaches. Each group contains six neighbourhood structures. The first two sets use a steepest descent and EMC local search whilst the third set uses a 3-opt operator. The solution returned by a local search, after exploring a neighbourhood structure, will be accepted based on the EMCQ acceptance criterion.

d) An on-line constructive heuristic and a novel weighted nozzle rank heuristic to optimise the component pick-and-place operations of the hybrid pick-and-place machine of a new SMD placement machine has been presented.

   This on-line scheduling approach is different from the one proposed in (e) in which it only allowed the machine to begin a pickup-and-placement operation once a complete schedule was available, prepared an improved schedule for the subsequent PCB only (not for the current PCB being processed), ignored the nozzle change operation (it assumed that all components can be picked up by the same nozzle) and modelled different types of machine specification. This new approach, (d), more closely mirrors the real-world, which previous works did not address in (e). The new theoretical on-line scheduler can construct an on-line schedule for the subsequent PCB points while the machine is performing pickup-and-placement operations. The theoretical on-line scheduling approach might continually produce a good schedule without incurring any significant cost even if there is a change in the resources after the schedule is generated (allowing the machine to run continually).

The weighted nozzle rank heuristic intelligently chooses a nozzle pair based on a new weighted nozzle rank procedure. The nozzle is ranked based on the effectiveness of the sub tour's operation type and the appropriate counter values of the nozzle pair. The larger the counter values of the nozzle pair, the more likely the nozzle pair is to be chosen. By using a weighted nozzle rank heuristic, the tradeoff issues of having many good quality sub tours or minimising the tool change operations might be eliminated.

e) A methodology for on-line scheduling to sequence the pickup-and-placement of components on a multi-head SMD placement machine has been proposed.

Due to the dynamic nature of the PCB assembly process, off-line scheduling is inefficient. For example, if the component feeders are misallocated by the machine's operator or if some components are missing from the feeder carrier (e.g. they run out), then the solution given by an off-line scheduler becomes infeasible. The proposed theoretical on-line scheduling methodology might overcome these spontaneous circumstances. Some of the latest technology in SMD placement machines are fitted with smart feeder carrier(s) that can automatically detect the availability and location of each component type on the feeder slot. This allows a feeder changeover while the machine is running and does not require a fixed feeder location (i.e. we can randomly place the component feeders in any feeder slot). Therefore, the proposed theoretical on-line scheduling methodology exploited this feature to enhance the scheduling of the PCB machine. Indeed, the proposed theoretical on-line scheduling might eliminate the machine's idling time by starting the pickup-and-placement operations immediately after the PCB (and the PCB data) have been loaded into the machine and the machine might continuously run even if there are missing components or a feeder changeover occurs.

9.4 Future Work

These investigations have identified some interesting directions for future research:

a) We suggest applying the improvement heuristics suggested in chapter 5, 6 and 7 to further improve the component pick-and-place sequence for the hybrid pick-and-place SMD placement machine.

   The work in chapter 8 only considered a constructive heuristic. This on-line scheduling can be integrated with any other meta-heuristic such as some that suggested in chapter 5, 6 and 7 to continually seek for better schedule quality while the machine is running.

b) It is interesting to consider tray feeders in optimising the component pick-and-place sequence for the hybrid pick-and-place SMD placement machine.

   The large size components supplied in trays are fed using tray feeders. Some machines allow a single tray to be placed into the machine feeding area whilst the others using an automatic tray-handling unit. A tray changer holds several platforms, that each can hold one or more trays. Trays are available only if the correct platform is loaded. Changing a platform is a very slow process that takes several seconds (e.g. 10 seconds for the HP-110). Therefore proper tray scheduling is a crucial decision in optimising the SMD placement machine throughput.

c) The use of multi-agent system techniques might be applicable in solving the problems associated with SMD placement machines.

   A multi-agent system (MAS) is a system that has many autonomous agents, which interact with each other to reach common objectives, whilst concurrently each agent pursues individual objectives (Ferber, 1999). As the optimisation of the SMD placement machine involve many intertwined sub-problem, applying MAS might be beneficial in enhancing the machine throughput. Moreover, this is an unexplored area in this field.

d) It might be suitable to apply genetic algorithms (GA) to the optimisation problems surrounding the hybrid pick-and-place SMD placement machines.

   GA's has been successfully applied to solve many optimisations of the SMD placement machine problems (Ho and Ji, 2003 and 2004; Jeevan et al., 2002; Khoo and Loh, 2000; Leu et al., 1993; Ong and Khoo, 1999; Wang et al. 1999). Therefore, utilising a GA approach might be helpful to tackle the optimisation problem of the hybrid pick-and-place SMD placement machine.

e) We would recommend applying the heuristics developed in this thesis to datasets drawn from the real world.

At present, the heuristic has only been tested on randomly generated datasets based on the problem description by DIMA's machine expert. As the experimental results from the proposed scheduler were very promising, the industrial partner (DIMA machine expert group) is happy to implement the proposed approach onto the real machine. To do so, requires some modifications to ensure correct communication between the scheduler and other software on the SMD placement machine.

REFERENCES

Aarts, E. H. L. (1989). Simulated annealing and Boltzmann machines: a stochastic approach to combinatorial optimization and neural computing. Wiley.

Aarts, E. and Lenstra, J. K. (eds). (1997). *Local search in combinatorial optimization*. Wiley.

Aarts, E. and Lenstra, J. K. (eds). (2003). *Local search in combinatorial optimization*. Princeton University Press.

Abdullah, S., Ahmadi, S., Burke, E. K. and Dror, M. (2004). Investigating Ahuja-Orlin's large neighbourhood search for examination timetabling. *Technical Report at School of Computer Science and Information Technology*, Nottingham University, NOTTCS-TR-2004-8 (submitted to OR Spectrum).

Adzakpa, K.P., Adjallah, K.H. and Yalaoui, F. (2004). On-line maintenance job scheduling and assignment to resources in distributed systems by heuristic-based optimization. *Journal of Intelligent Manufacturing*, 15(2), 131-140.

Ahmadi, J., Grotzinger, S. and Johnson, D. (1988). Component allocation and partitioning for a dual delivery placement machine, *Operations Research*, 36, 176-191.

Ahmadi, J., Grotzinger, S. and Johnson, D. (1991). Emulating concurrency in a circuit card assembly system. *International Journal of Flexible Manufacturing Systems*, 3(1), 45-70.

Ahmadi, R. H. (1993). *A hierarchical approach to design, planning, and control problems in electronic circuit card manufacturing*. Perspectives in Operations Management, Sarin, R.K. editor, Kluwer Academic Publishers, Dordrecht, 409-429.

Ahmadi, J., Ahmadi, R., Matsuo, H. and Tirupati, D. (1995). Component fixture positioning for printed circuit board assembly with concurrent operations. *Operations Research*, 43, 444-457.

Ahmadi, R. H. and Mamer, J. W. (1999). Routing heuristics for automated pick and place machines. *European Journal of Operational Research*, 117, 533-552.

Aickelin, U. and Dowsland, K.A. (2000). Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem. *Journal of Scheduling*, 3, 139-153.

Altinkemer, K., Kazaz, B., Koksalan, M. and Moskowitz, H. (2000). Optimization of printed circuit board manufacturing: Integrated modeling and algorithms. *European Journal of Operational Research*, 124, 409-421.

Avanthay, C., Hertz, A. and Zufferey, N. (2003). A variable neighborhood search for graph coloring. *European Journal of Operational Research*, 151, 379-388.

Ayob, M., Cowling, P. and Kendall, G.(2002) Optimisation for surface mount placement machines. *Proceedings of the IEEE ICIT'02*, Bangkok, 11-14 December 2002, 498-503.

Ayob, M. and Kendall, G. (2002). A new dynamic point specification approach to optimise surface mount placement machine in printed circuit board assembly. *Proceedings of the IEEE ICIT'02*, Bangkok, 11-14 December, 486-491.

Ayob, M. and Kendall, G. (2003a). Real-time scheduling for multi headed placement machine. *Proceedings of the 5th IEEE International Symposium on Assembly and Task Planning, ISATP'03*, Besançom, France, 9-11 July, 128-133.

Ayob, M. and Kendall, G. (2003b). An investigation of an adaptive scheduling for multi headed placement machines. *Proceedings of the 1st Multidisciplinary International Conference on Scheduling: Theory and Applications, MISTA 2003*, Nottingham, UK, 13-16 August, 363-380.

Ayob, M. and Kendall, G. (2003c). A monte carlo hyper-heuristic to optimise component placement sequencing for multi head placement machine. *Proceedings of the International Conference on Intelligent Technologies, InTech'03*, Chiang Mai, Thailand, 17-19 Dec, 132-141.

Ayob, M. and Kendall, G. (2004). A nozzle selection heuristic to optimise the hybrid pick and place machine. *Proceedings of the 2004 IEEE Conference on Cybernetics and Intelligent Systems (CIS 2004)*, Singapore, 1259-1264.

Ayob, M. and Kendall, G. (2005a). A triple objective function with a chebychev dynamic point specification approach to optimise the SMD placement machine. *European Journal of Operational Research,* 164, 609-626.

Ayob M. and Kendall, G. (2005b). A variable neighbourhood monte carlo search for component placement sequencing of multi headed placement machine in printed circuit board assembly. Submitted to the *Intelligent Manufacturing Journal*.

Ayob M. and Kendall, G. (2005c). A weighted nozzle rank heuristic to optimise the hybrid pick and place machine. Submitted to the *International Journal of Production Research*.

Bai, R. and Kendall, G. (2003). An investigation of automated planograms using a simulated annealing based hyper-heuristics. *Proceedings of the Fifth Metaheuristics International Conference (MIC 2003)*, Kyoto International Conference Hall, Kyoto, Japan, 23-25 August. To appear in the selected volume of the conference, Ibaraki T., Nonobe K and Yagiura M. (eds), 2005.

Ball, M. O. and Magazine M. J. (1988). Sequencing of insertions in printed circuit board assembly. *Operations Research*, 36, 192-201.

Bard, J. F. (1988). A heuristic for minimizing the number of tool switches on a flexible machine. *IIE Transactions* 20, 382--391.

Bard, J. F., Clayton, R. W. and Feo, T.A. (1994). Machine setup and component placement in printed circuit board assembly. *International Journal of Flexible Manufacturing Systems*, 6, 5-31.

Bentzen, B. (2000). SMD placement, in the *SMT in FOCUS*. url:*http://www.smtinfocus.com/PDF/SMD_placement.pdf* (Sept. 25, 2002).

Blum, C. and Roli, A. (2001). Metaheuristics in combinatorial optimization: overview and conceptual comparison. *Technical Report TR/IRIDIA/2001-13*, IRIDIA, Belgium.

Bonert, M., Shu, L. H. and Benhabib, B. (2000). Motion planning for multi-robot assembly systems. *International Journal of Computer Integrated Manufacturing*, 13(4), 301-310.

Burke, E. K., Cowling, P. I. and Keuthen, R. (1999). New models and heuristics for component placement in printed circuit board assembly. *Proceedings of the 1999 IEEE on ICIIS99*, 33-140.

Burke, E. K., Cowling, P. and Keuthen, R. (2001). The printed circuit board assembly problem: heuristic approaches for multi-headed placement machinery. *Proceedings of the IC-AI2001*, Las Vegas, CSREA Press, 1456-1462.

Burke, E. K., MacCarthy, B., Petrovic, S. and Qu, R. (2002). Knowledge discovery in hyper-heuristic using case-based reasoning on course timetabling. *Selected Papers from the PATAT'02*, LNCS 2740, 276-287.

Burke, E., Hart, E., Kendall, G., Newall, J., Ross, P. and Schulenburg, S. (2003a). *Hyper-Heuristics: An emerging direction in modern search technology. ch. 16*. In: Glover, F. and Kochenberger, G. (eds) Handbook of Meta-Heuristics. Kluwer, 457-474.

Burke, E. K., Kendall, G. and Soubeiga, E. (2003b). A tabu-search hyper-heuristic for timetabling and rostering. *Journal of Heuristics*, 9 (6), 451-470.

Burke, E. K., Landa Silva, J. D. and Soubeiga, E. (2003c). Hyperheuristic approaches for multiobjective optimisation. *Proceedings of the Fifth Metaheuristics International Conference (MIC 2003)*, Kyoto Japan, August 25-28.

Burke, E. K., Dror, M., Petrovic, S. and Qu, R. (2005a) *Hybrid graph heuristics within a hyper-heuristic approach to exam timetabling problems*. In Golden, B. L., Raghavan, S. and Wasil, E. A. (eds.). The Next Wave in Computing, Optimization, and Decision Technologies. Conference Volume of the 9th informs Computing Society Conference. Springer, 79-91.

Burke, E. K., Petrovic, S. and Qu, R. (2005b). Case based heuristic selection for timetabling problems. *Journal of Scheduling*, forthcoming.

Caporossi, G., Gutman, I. and Hansen, P. (1999). Variable neighborhood search for extremal graphs IV: Chemical trees with extremal connectivity index. *Computers and Chemistry*, 23, 469-477.

Caporossi, G. and Hansen, P. (2000). Variable neighborhood search for extremal graphs: I The AutoGraphiX system. *Discrete Mathematics*, 212, 29-44.

Caporossi, G. and Hansen, P. (2004). Variable neighborhood search for extremal graphs 5. Three ways to automate finding conjectures. *Discrete Mathematics*, 276, 81-94.

Cavalloro, P. and Cividadi, E. (1988). An expert system for process planning in PCB assembly line. *CH2552-8/88/0000/0170, IEEE*, 170-174.

Chan, D. (1993). Precedence constrained TSP applied to circuit board assembly and no wait flow-shop. *International Journal of Production Research*, 31(9), 2171-2177.

Chan, D. and Mercier, D. (1989). IC insertion: an application of the traveling salesman problem. *International Journal of Production Research*, 27, 1837-1841.

Chandra, P., Li, S. and Stan, M. (1993). Jobs and tool sequencing in an automated manufacturing environment. *International Journal of Production Research*, 31, 2911-2925.

Chang, T. C. and Terwilliger, J. (1987). Rule-based system for printed wiring assembly process planning. *International Journal of Production Research*, 25, 1465-1482.

Chen, W-S. and Chyu, C-C. (2003). A minimum setup strategy for sequencing PCBs with multi-slot feeders. *Integrated Manufacturing Systems*, 14(3), 255-267.

Chiu, C., Yih, Y. and Chang, T. C. (1991). A collision-free sequencing algorithm for PWB assembly. *Journal of Electronics Manufacturing*, 1(1), 1-12.

Cowling, P. and Johansson, M. (2002). Using real-time information for effective dynamic scheduling. *European Journal of Operational Research*, 139, 230-244.

Cowling, P., Kendall, G. and Soubeiga, E., (2001a). A parameter free hyperheuristic for scheduling a sales summit. *Proceedings of the Third Metaheuristic International Conference, MIC 2001*, 127-131.

Cowling, P., Kendall, G. and Soubeiga, E. (2001b). A hyperheuristic approach to scheduling a sales summit. In Burke, E. and Erben, W., editors, *Selected Papers Of The Third International Conference On The Practice And Theory Of Automated Timetabling, PATAT'2000*, Springer Lecture Notes in Computer Science, 176-190.

Cowling, P., Kendall, G. and Han, L. (2002a). An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem. *Proceedings of Congress on Evolutionary Computation, CEC2002*, Hawaii, May 12-17, 1185-1190.

Cowling, P., Kendall, G. and Soubeiga, E. (2002b) Hyperheuristics: A tool for rapid prototyping in scheduling and optimisation. *Proceedings of 2nd European Workshop on Evolutionary Computation in Combinatorial Optimisation, EvoCOP2002*, Springer LNCS, 1-10.

Cowling, P., Kendall, G. and Soubeiga, E. (2002c). Hyperheuristics: A robust optimisation method applied to nurse scheduling. *Proceedings of 7th International Conference on Parallel Problem Solving from Nature*, Springer, LNCS, 851-860.

Crainic, T.G., Gendreau, M., Hansen, P. and Mladenović, N. (2004). Cooperative parallel variable neighborhood search for the p-median. *Journal of Heuristics*, 10(3), 293-314.

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.

Crama, Y., Kolen, A. W. J., Oerlemans, A. G. and Spieksma, F. C. R. (1994). Minimizing the number of tool switches on a flexible machine. *International Journal of Flexible Manufacturing Systems*, 6, 33-54.

Crama, Y., Flippo, O.E., van de Klundert, J. J. and Spieksma, F. C. R. (1996). The component retrieval problem in printed circuit board assembly. *International Journal of Flexible Manufacturing Systems*, 8, 287-312.

Crama, Y., Flippo, O. E., Klundert, J. 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.

Crama, Y., Klundert, J. van de and Spieksma, F. C. R. (2002). Production planning problems in printed circuit board assembly. *Discrete Applied Mathematics*, 123, 339-361.

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.

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 System, Man, and Cybernetics-Part C: Applications and Reviews*, 30(4), 408-417.

Deo, S., Javadpour, R. and Knapp, G. M. (2002). Multiple setup PCB assembly planning using genetic algorithms. *Computers & Industrial Engineering*, 42, 1-16.

DePuy, G. W., Ammons, J. C. and McGinnis, L. F. (2000). Multiple assignment of component types to feeder slots on automated printed circuit card placement machines. *IEEE Transactions on Electronics Packaging Manufacturing*, 23(3), 157-164.

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.

De Souza, R. and Lijun, W. (1995). Intelligent optimization of component insertion in multi-head concurrent operation PCBA machines. *Journal of Intelligent Manufacturing*, 6, 235-243.

Dikos, A., Nelson, P.C., Tirpak, T.M. and Wang, W. (1997). Optimization of high-mix printed circuit card assembly using genetic algorithms. *Annals of Operations Research*, 75, 303-324.

Dima SMT Systems. (2003). Hybrid P&P HP-110, *User Manual Version 1.2*.

Dowsland, K. A. (1995). *Simulated Annealing, ch. 2*. In: Reeves, C. R. (eds) Modern heuristic techniques for combinatorial problems, McGraw-Hill, pp 20-69.

Dowsland, K., Soubeiga, E., Burke, E. (2005a). Solving a shipper rationalisation problem with a simulated annealing hyperheuristic. *European Journal of Operational Research*, forthcoming.

Dowsland, K., Soubeiga, E., Burke, E. (2005b). A simulated annealing hyper-heuristic for determining shipper sizes. *European Journal of Operational Research*, forthcoming.

Drezner, Z. and Nof, S. (1984). On optimizing bin picking and insertion plans for assembly robots. *IIE Transactions*, 16, 262-270.

Duman, E and Or, I. (2004) Precedence constrained TSP arising in printed circuit board assembly. *International Journal of Production Research*, 42(1), 67-78.

Ellis, K. P, Vittes, F. J. and Kobza, J. E. (2001). Optimizing the performance of a surface mount placement machine. *IEEE Transactions on Electronic Packaging Manufacturing*, 24(3), 160-170.

Ellis, K. P., Kobza, J. E. and Vittes, F. J. (2002). Development of placement time estimator function for a turret style surface mount placement machine, *Robotic and Computer Integrated Manufacturing*, 18, 241-254.

Ferber, J. (1999). *Multi-agent systems: An introduction to distributed artificial intelligence*. Addison-Wesley, London.

Fleszar, K. and Hindi, K. H. (2002) New heuristics for one-dimensional bin-packing. *Computers and Operations Research*, 29, 821-839.

Fleszar, K. and Hindi, K. H. (2004). Solving the resource-constrained project scheduling problem by a variable neighbourhood search. *European Journal of Operational Research*, 155(2), 402-413.

Fisher, H. and Thompson, G. L. (1963). *Probabilistic learning combinations of local job-shop scheduling rules*. In Muth, J.F. and Thompson, G.L. (eds), Industrial Scheduling, New Jersey:Prentice-Hall, Inc., 225-251.

Foulds, L. R. (1983). The heuristic problem-solving approach. *JORS*, 34, 927-934.

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.

Francis, R. L., Hamacher, H. W., Lee, C. -Y. and Yeralan, S. (1994). Finding placement sequences and bin locations for cartesian robots. *IIE Transactions*, 26, 47-59.

Francis, R. L., McGinnis, L. F. and White, J. A. (1992). *Facility Layout and Location: An Analytical Approach*. Englewood Cliffs, NJ:Prentice-Hall.

Fu, H. –P. and Su, C. –T. (2000). A comparison of search techniques for minimizing assembly time in Printed Wiring Assembly. *Journal of Production Economics*, 63, 83-98.

Fuji Product catalog, 2001.

Galinier, P. Hao, J. -K. (1999). Hybrid evolutionary algorithms for graph coloring. *Journal. of Combinatorial Optimization*, 3, 379-397.

Garey, M. R. and Johnson, D. S. (1979). *Computers and intractability-a guide to the theory of NP-completeness*. Freeman.

Gastel, S. V. (2002). A comparison of SMD placement machine concepts, in the *SMT in FOCUS*. url:*http://www.smtinfocus.com/PDF/SMD_machine_concept_ Assembleon.pdf* (Oct. 6, 2004).

Gavish, B. and Seidmann, A. (1988). Printed circuit boards assembly automation-formulations and algorithms. In Mital, A. (eds) *Recent Developments in Production Research*, Amsterdam: Elsevier Science, 624-635.

Gaw, A., Rattadilok, P. and Kwan, R. S. K. (2004). Distributed choice function hyperheuristics for timetabling and scheduling. In Burke, E. and Trick M. (eds) *Proceedings of the 5th international conference on the Practice and Theory of Automated Timetabling (PATAT)*, Pittsburgh, USA, 18-20 Aug, 495-497.

Glover, F. (1989). Tabu search: part I. *ORSA Journal on Computing*, 1, 190-206.

Glover, F. (1990). Tabu search: part II. *ORSA Journal on Computing*, 2, 4-32.

Glover, F. and Laguna, M. (1995). Tabu search, ch. 3. In Reeves, C. R.(eds) *Modern heuristic techniques for combinatorial problems*, McGraw-Hill, pp 70-150.

Glover, F. and Laguna, M. (1997). *Tabu search*. Kluwer Academic Publishers.

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

Goldberg, D. E. (1989). *Genetic algorithms in search, optimisation and machine learning*. Addison Wesley.

Grotzinger, S. (1992). Feeder assignment models for concurrent placement machines. *IIE Transactions*, 24, 31-46.

Grotzinger, S. and Sciomachen, A. (1988). A petri net characterization of a high speed placement machine. *Proceedings of the 38th Electronic components Conference*, Los Angeles, California, 64-68.

Han, L. and Kendall, G. (2003). Investigation of a tabu assisted hyper-heuristic genetic algorithm. *Proceedings of Congress on Evolutionary Computation (CEC2003)*, Canberra, Australia, Dec 8-12, Vol. 3, 2230-2237.

Hansen, P. and Mladenović, N. (1997). Variable neighborhood search for the P-median. *Location Science*, 5(4), 207-226.

Hansen, P., Mladenović, N. and Perez-Britos, D. (2001) Variable neighborhood decomposition search. *Journal of Heuristics*, 7, 335-350.

Hansen, P. and Mladenović, N. (2001). Variable neighborhood search. *European Journal of Operational Research*, 130, 449-467.

Hart, E. and Ross, P. (1998). A heuristic combination method for solving job-shop scheduling problems, in Eiben, A.E., Back, T., Schoenauer, M. and Schwefel, H.P., (eds), Parallel problem solving from nature V, *Lecture Notes in Computer Science*, 1498, Springer-Verlag, 845-854.

Hart, E., Ross, P. and Nelson, J. A. (1998). Solving a real-world problem using an evolving heuristically driven schedule builder. *Evolutionary Computing*, 6(1), 61-80.

Hertz, A. de Werra, D. (1987). Tabu search techniques for graph coloring. *Computing*, 39, 345-351.

Ho, W. and Ji, P. (2003) Component scheduling for chip shooter machines: a hybrid genetic algorithm approach. *Computers and Operations Research*, 30, 2175-2189.

Ho, W. and Ji, P. (2004). A hybrid genetic algorithm for component sequencing and feeder arrangement. *Journal of Intelligent Manufacturing*, 15(3), 307-315.

Hop, N.V and Tabucanon, M. T. (2001a). Multiple criteria approach for solving feeder assignment and assembly sequence problem in PCB assembly. *Journal of Production Planning and Control*, 12(8), 736-744.

Hop, N.V and Tabucanon, M. T. (2001b). Extended dynamic point specification approach to sequencing robot moves for PCB assembly. *International Journal of Production Research*, 39(8), 1671-1687.

Huang, Y. W. and Srihari, K. (1993). A solution methodology for the multiple batch surface mount PCB placement sequence problem. *Advances in Electronic Packaging American Society of Mechanical Engineers*, 373-379.

Jeevan, K., Parthiban, A., Seetharamu, K. N., Azid, I. A. and Quadir, G. A. (2002). Optimization of PCB Component Placement using Genetic Algorithms. *Journal of Electronics Manufacturing*, 11(1), 69-79.

Ji, P., Wong, Y. S., Loh, H. T. and Lee, L. C. (1994). SMT production scheduling: a generalized transportation approach. *International Journal of Production Research*, 32(10), 2323-2333.

Ji, P. and Wan, Y. F. (2001). Planning for printed circuit board assembly: the state-of-art review. *International Journal of Computer Applications in Technology*, 14, 136-144.

Ji, Z. Leu, M. C. and Wong, H. (1992). Application of linear assignment model for planning of robotic PC board assembly. *ASME Journal of Electronic Packaging*, 114, 455-460.

Johnson, D. S. (1990). Local optimization and the traveling salesman problem. In Goos, G. and Hartmanis, J. (eds) *Automata, Languages and Programming*, Lecture Notes in Computer Science, 442, Springer, Heidelberg, 446-461.

Kang, M.-J. and Han, C.-G. (1998). Solving the rural postman problem using a genetic algorithm with a graph transformation. *Proceedings of the ACM symposium on Applied Computing,* Georgia, USA, February, 356-360.

Karg, R. and Thompson, G. L. (1964). A heuristic approach to solving traveling salesman problems, *Management Science*, 10.

Kazaz, B. and Altinkemer, K. (2003). Optimization of multi-feeder (depot) printed circuit board manufacturing with error guarantees. *European Journal of Operational Research*, 150(2), 370-394.

Kendall, G. and Mohamad, M. (2004a). Solving the fixed channel assignment problem in cellular communications using an adaptive local search. *Proceedings of the 5th international conference on the Practice and Theory of Automated Timetabling (PATAT)*, Pittsburgh, USA, 18 - 20 August, 219-231.

Kendall, G. and Mohamad, M. (2004b). Channel assignment in cellular communication using a great deluge hyper-heuristic. *Proceedings of the 2004 IEEE International Conference on Network (ICON2004)*, Singapore, 16-19 November, 769-773.

Kendall, G. and Mohamad, M. (2004c). Channel assignment optimisation using a hyper-heuristic. *Proceedings of the 2004 IEEE Conference on Cybernetics and Intelligent Systems (CIS)*, Singapore, 1-3 December, 790-795.

Kendall, G. and Mohd Hussin, N. (2003). An investigation of a tabu search based hyper-heuristic for examination timetabling. *Proceedings of the 1st Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA2003)*, 226-233. To appear in selected papers from MISTA 2003, Kendall G., Burke E. and Petrovic S. (eds), Kluwer Publication, 2005.

Kendall G. and Mohd Hussin, N. (2004). Tabu search hyper-heuristic approach to the examination timetabling problem at University Technology MARA. In: Burke, E. and Trick M. (eds): *Proceedings of the 5th international conference on the Practice and Theory of Automated Timetabling (PATAT)*, Pittsburgh, USA, 18-20 Aug, 199-217.

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(4), 289-296.

Khoo, L.P. and Ng, T.K. (1998). A genetic algorithm-based planning system for PCB component placement. *International Journal of Production Economics*, 54, 321-332.

Khoo, L.P. and Ong, N. S. (1998). PCB assembly planning using genetic algorithms. *International Journal of Advanced Manufacturing Technology*, 14, 363-368.

Klincewicz, J.G. and Rajan, A. (1994). Using GRASP to solve the component grouping problem. *Naval Research Logistics*, 41,893-912.

Klomp, C., Klundert, J. 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.

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

Kumar.R. and Luo. Z. (2003). Optimizing the operation sequence of a chip placement machine using TSP model. *IEEE Transactions on Electronics Packaging Manufacturing*, 26(1), 14-21.

Kutanoglu, E. and Sabuncuoglu, I. (2001). Experimental investigation of iterative simulation-based scheduling in a dynamic and stochastic job shop. *Journal of Manufacturing Systems*, 20(4), 264-279.

Lee, S. H., Lee, B. H. and Park, T. H. (1999). A hierarchical method to improve the productivity of a multi-head surface mounting machine. *Proceedings of the 1999 IEEE on Robotics and Automation*, Michigan, 2110-2115.

Leipälä, T. and Nevalainen, O. (1989). Optimization of the movements of a component placement machine. *European Journal of Operational Research*, 38, 167-177.

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.

Liggett, R. S. (1981). The quadratic assignment problem: an experimental investigation of solution strategies. *Management Science*, 27(4), 442-458.

Loh, T. S, Bukkapatnam, S. T. S., Medeiros, D. and Kwon, H. A. (2001). Genetic algorithm for sequential part assignment for PCB assembly. *Computers and Industrial Engineering*, 40, 293-307.

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.

Man, K. F., Tang, K. S. and Kwong, S. (1999). *Genetic algorithm: concepts and design*. Springer.

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 Transactions: Scheduling & Logistics,* 24, 18-30.

Mettalla, E. G. and Egeblu, P. J. (1989). Alternative Approaches To Sequencing Robot Moves For PCB Assembly. *International Journal of Computer Integrated Manufacturing*, 2, 243-256.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers and Operations Research,* 24(11), 1097-1100.

Mladenović, N., Petrović, Kovačević-Vujčić, V. and Čangalović, M. (2003a). Solving spread spectrum radar polyphase code design problem by tabu search and variable neighbourhood search. *European Journal of Operational Research*, 151, 389-399.

Mladenović, N., Labbé, M. and Hansen, P. (2003b). Solving the p-Center problem with tabu search and variable neighborhood search. *Networks*, 42(1), 48-64.

Morena Pérez, J.A., Marcos Moreno-Vega, J. and Rodríguez Martín, I. (2003). Variable neighborhood tabu search and its application to the median cycle problem. *European Journal of Operational Research*, 151, 365-378.

Moyer, L.K. and Gupta, S. M. (1996a). SMT feeder slot assignment for predetermined component placement paths. *Journal of Electronics Manufacturing*, 6, 173-192.

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.

Moyer, L.K. and Gupta, S. M. (1997). An efficient assembly sequencing heuristic for printed circuit board configurations. *Journal of Electronics Manufacturing*, 7, 143-160.

Moyer, L.K. and Gupta, S. M. (1998). Development of the surface mount assembly process through an angular board orientation. *International Journal of Production Research*, 36, 1857-1881.

MYDATA automation worldwide brochure, (2002).

Nareyek, A. (2003). *Choosing search heuristics by non-stationary reinforcement learning*. In Pinho de Sousa, J. (eds), Metaheuristics:Computer Decision Making, Kluwer.

Nelson, K. M. and Wille, L.T. (1995). Comparative study of heuristics for optimal printed circuit board assembly. *Proceedings of Southcon'95*. Fort Lauderdale, FLorida, USA, 322-327.

Ng, M. (1998). Heuristics approach to printed circuit board insertion problem. *Journal of the Operational Resarch Society*, 49, 1051-1059.

Ng, M. K. (2000). Clustering methods for printed circuit board insertion problems, *Journal of the Operational Research Society*, 51(10), 1205-1211.

Ohno, K., Jin, Z. and Elmaghraby, S. E. (1999). An optimal assembly mode of multi-type printed circuit boards. *Computers and Industrial Engineering*, 36, 451-471.

Ong, N-S. and Khoo, L.P. (1999). Genetic algorithm approach in PCB assembly. *Integrated Manufacturing Systems*, 10(5), 256-265.

Ong, N. S. and Tan, W. C. (2002). Sequence placement planning for high-speed PCB assembly machine. *Integrated Manufacturing Systems*, 13(1), 13, 35-45.

Osman, I. H. (1996). Meta-Heuristics: An overview. In: Ibrahim, I.H. and Kelly, J.P.(eds) *Meta-Heuristics: Theory & applications*, Kluwer Academic Publishers, 1-21.

Ouelhadj, D. (2003). A Multi-Agent system for the integrated dynamic scheduling of steel production, *PhD Thesis*, School of Computer Science and Information Technology, Nottingham University.

Ovacik, I.M. and Uzsoy, R. (1994). Rolling horizon algorithms for a single machine dynamic scheduling problem with sequence dependent setup times. *International Journal of Production Research*, 32(6), 1243-1263.

Panasonic Electronic Component Assembly System, Brochure, 2001.

Pearn, W. L. and Wu, T. C. (1995). Algorithm for the rural postman problem. *Computers and Operations Research*, 22(8), 819-828.

Petrovic, S. and Qu, R. (2002). Case-based reasoning as a heuristic selector in a hyper-heuristic for course timetabling. *Proceedings of the 6th International Conference on Knowledge-Based Intelligent Information & Engineering Systems and Applied Technologies, KES 2002*, Milan, Italy, Vol. 82, 336-340.

Polacek, M., Hartl, R.F., Doerner, K. and Reimann, M. (2004). A variable neighborhood search for the multi depot vehicle routing problem with time windows. *Journal of Heuristics*, 10(6), 613-627.

Rayward-Smith, V. J. (1986). *A first course in computability*. Blackwell.

Reeves, C.R. and Beasley, J.E. (1995). *Introduction, ch. 1*. In: Reeves, C. R.(eds) Modern heuristic techniques for combinatorial problems, McGraw-Hill, 1-19.

Ross, P., Hart, E., Burke, E., Cowling, P., Kendall, G. and Petrovic, S. (2000). An investigation of hyper-heuristic methods, EPSRC case for support, project number GR/N36837/01, url:*http://www.dcs.napier.ac.uk/~sonias/proposal.ps.*

Ross, P., Schulenburg, S., Marín-Blázquez, J.G. and Hart, E. (2002). Hyper-heuristics: Learning to combine simple heuristics in bin-packing problem. *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2002*, Morgan Kauffmann, 942-948.

Ross, P., Marín-Blázquez, J. G., Schulenburg, S and Hart, E. (2003). Learning a procedure that can solve hard bin-packing problems: A new GA-based approach to hyper-heuristics. *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2003,* Berlin, Germany, 1295-1306.

Sabuncuoglu, I. and Bayiz, M. (2000). Analysis of reactive scheduling problems in a job shop environment. *European Journal of Operational Research*, 126(3), 567-586.

Sabuncuoglu, I. And Karabuk, S. (1999). Rescheduling frequency in an FMS with uncertain processing times and unreliable machines. *Journal of Manufacturing Systems*, 18(4), 268-283.

Sadiq, M., Landers, L. and Don Taylor, G. (1993). A heuristic algorithm for minimizing total production time for a sequence of jobs on a surface mount placement machine. *International Journal of Production Research*, 31(6), 1327-1341.

Safai, F. (1996). Cycle time improvement for Fuji IP2 pick-and-place machines. *Hewlett-Packard Journal*, 47(4), 80-86.

Samsung electronics: robots. (2001). URL: *http://www.samsungelectronics.com/factory_automation/robots/*, Samsung Electronics Co., Ltd.

Sanchez, J. M. and Priest, J. W. (1991). Optimal component insertation sequence planning methodology for the semi-automatic assembly PCB. *Journal of Intelligent Manufacturing*, 2, 177-188.

Schulenburg, S, Ross, P., Marín-Blázquez, J.G. and Hart, E. (2002). A hyper-heuristic approach to single and multiple step environments in bin-packing problems. *Proceedings of the Fifth International Workshop on Learning Classifier Systems 2002*, IWLCS-02.

Shakeri, M. (2004). Implementation of an automated operation planning and optimum operation sequencing and tool selection algorithms. *Computers in Industry,* 54(3), 2004, 223-236.

Shih, W., Srihari, K. and Adriance, J. (1996). Expert system based placement sequence identification for surface mount PCB assembly. *International Journal of Advanced Manufacturing Technology*, 11, 413-424.

Silberschatz, A., Galvin, P. and Gagne, G. (2000). *Applied operating system concepts*. John Wiley & Sons, USA.

Silver, E., Vidal, R. V. and de Werra, D. (1980). A tutorial on heuristic methods. *European Journal of Operational Research*, 5, 153-162.

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.

Soubeiga, E. (2003). Development and application of hyper-heuristics to personnel scheduling, *PhD Thesis*, School of Computer Science and Information Technology, Nottingham University.

Srinivasan, K. and Sanii, E. T. (1991). AI-based process planning for electronic assembly. *IIE Transactions*, 23(2), 127-137.

Su, C. and Fu, H. (1998). A simulated annealing heuristic for robotics assembly using the dynamic pick-and-place model. *Production Planning and Control*, 9, 795-802.

Su, C., Ho, L. and Fu, H. (1998). A novel tabu search approach to find the best placement sequence and magazine assignment in dynamic robotics assembly. *Integrated Manufacturing Systems*, 9, 366-376.

Su, Y. and Srihari, K. (1996). Placement sequence identification using artificial neural networks in surface mount PCB assembly. *International Journal of Advanced Manufacturing Technology*, 11, 285-299.

Su, Y. -C., Wang, C., Egbelu, P. J. and Cannon, D. J. (1995). A dynamic point specification approach to sequencing robot moves for PCB assembly. *International Journal of Computer integrated Manufacturing*, 8(6), 448-456.

Sule, D. R. (1993). Job sequencing to minimize tool changeovers in flexible manufacturing systems. Production *Planning & Control*, 4, 46-53.

Sun, D. -S., Lee, T. -E and Kim, K. -H. (2004). Component allocation and feeder arrangement for a dual-gantry multi-head surface mounting placement tool. *International Journal of Production Economics*, in press.

Sun, J. and Xue, D. (2001). A dynamic reactive scheduling mechanism for responding to changes of production orders and manufacturing resources. *Computers in Industry*, 46, 189-207.

Tanenbaum, A. (1992). *Modern operating systems*. Prentice-Hall, New Jersey, USA.

Tang, C. S. And Denardo, E. (1988a). Models arising from a flexible manufacturing machine, part 1: minimisation of the number of tool switches. *Operation Research*. 36, 767-777.

Tang, C. S. And Denardo, E. (1988b). Models arising from a flexible manufacturing machine, part II: minimisation of the number of tool switches. *Operation Research*. 36, 778-784.

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 Sympoium*, Twenty-Sixth IEEE/CPMT, 214-220.

Tirpak, T. M. (2000). Design to manufacturing information management for electronics assembly. *International Journal of Flexible Manufacturing Systems*, 12, 189-205.

Truss, J. K. (1999). *Discrete mathematics for computer scientists*. Addison-Wesley.

Universal Instruments Corporation. (1999). General surface mounter-the universal solution.URL:http://www.uic.com.

Van Laarhoven, P. J. M., and Zijm, W. H. M. (1993). Production preparation and numerical control in PCB assembly. *International Journal of Flexible Manufacturing Systems*, 5, 187-207.

Vieira, G.E., Hermann, J.W. and Lin, E. (2003). Rescheduling manufacturing systems: a framework of strategies, policies and methods. *Journal of Scheduling*, 6(1), 36-92.

Voss, S., Martello, S., Osman, I. H. and Roucairol, C. (eds). (1999). *Meta-heuristics: advances and trends in local search paradigms for optimization*, Kluwer Academic Publishers.

Wang, C. (1996). Layout designs for robotic PCB assembly. *Journal of Integrated Manufacturing Systems*, 7(4), 39–52.

Wang, C., Ho, L. S., Fu, H. P and Su, Y. C. (1995). A Magazine assignment heuristic for robotic assembly using the dynamic pick-and-place approach. *Report No. IM84J05*, *Department of Industrial Management*, Chung-Hua Polytechnic Institute, Hsinchu, Taiwan.

Wang, C., Ho, L. -S. and Cannon, D. J. (1998). Heuristics for assembly sequencing and relative magazine assignment for robotic assembly. *Computers and Industrial Engineering*, 34, 423-431.

Wang, C., Ho, L. S., Fu, H. P and Su, Y. C. (1997). A Magazine assignment heuristic for robotic assembly using the dynamic pick and place approach. *International Journal of Industrial Engineering*, 4(1), 24-33.

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.

Wong, H. and Leu, M. C. (1993). Adaptive genetic algorithm for optimal printed circuit board assembly planning. *Annals CIRP*, 42(1), 17-20.

Yeo, S. H., Low, C. W. Yong, K. H. (1996). A rule-based frame system for concurrent assembly machines. *International Journal of Advanced Manufacturing Technology*, 12, 370-376.

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.

APPENDIX A

Dataset N80K20_A: PCB points

N=80, K=20

Component ID	X	Y	Component type
1	172.25	170.50	1
2	122.75	172.00	2
3	80.38	80.63	7
4	557.25	172.00	7
5	159.25	90.13	2
6	490.50	55.25	17
7	400.75	161.25	16
8	88.25	139.00	5
9	491.38	156.00	15
10	120.13	88.88	6
11	83.75	183.75	18
12	350.75	61.75	8
13	435.63	98.63	18
14	439.50	190.88	15
15	578.00	86.13	14
16	180.50	37.25	9
17	554.88	193.13	12
18	438.25	52.63	12
19	337.63	76.50	8
20	327.25	131.38	1
21	576.13	186.63	9
22	185.38	165.88	19
23	564.25	190.88	9
24	306.75	116.63	16
25	84.25	190.00	18
26	567.50	113.00	17
27	434.50	63.75	12
28	512.13	197.88	5
29	292.75	126.75	12
30	401.00	73.00	2
31	260.50	58.50	11
32	385.13	17.13	8
33	355.13	134.50	12
34	346.13	51.50	15
35	81.75	110.88	20
36	120.63	181.75	11
37	18.88	35.88	1
38	202.38	40.50	19
39	81.25	187.38	13
40	262.13	33.88	20
41	358.00	118.38	16
42	184.88	129.38	9
43	439.25	84.88	14

Component ID	X	Y	Component type
44	58.25	107.75	19
45	187.25	112.38	12
46	377.88	81.88	20
47	12.13	130.00	4
48	462.63	52.25	19
49	518.63	93.63	1
50	356.38	132.50	6
51	186.25	78.63	13
52	352.38	182.25	16
53	134.75	48.75	1
54	541.88	164.50	10
55	213.75	64.00	1
56	246.75	48.13	13
57	430.75	36.88	6
58	336.00	195.25	17
59	78.38	160.13	3
60	280.38	141.88	15
61	99.13	108.50	8
62	115.63	107.38	20
63	92.25	73.75	5
64	582.13	199.75	10
65	360.00	84.13	13
66	372.75	190.38	3
67	547.50	171.88	1
68	231.88	149.75	14
69	251.38	62.63	5
70	336.50	116.75	14
71	33.75	167.50	17
72	173.13	193.13	4
73	461.75	179.63	7
74	281.88	41.75	4
75	372.00	198.00	6
76	325.13	173.13	8
77	412.13	46.75	19
78	528.38	179.75	17
79	550.50	33.63	2
80	221.50	58.75	11

Dataset N80K20_A: Feeder setup

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

Dataset N240K40_F: PCB points

N=240, K=40

Component ID	X	Y	Component type
1	926.63	261.38	27
2	963.38	54.88	32
3	1227.50	552.13	22
4	103.50	80.75	19
5	1762.38	14.25	6
6	1272.25	578.00	11
7	231.13	355.75	2
8	698.00	144.00	31
9	595.88	398.25	2
10	1168.00	513.38	34
11	783.25	81.75	31
12	7.25	25.25	3
13	1242.25	360.25	37
14	525.38	80.88	25
15	307.63	200.75	19
16	897.25	85.88	27
17	480.00	400.25	25
18	1422.50	277.88	2
19	390.38	21.50	1
20	308.13	493.63	11
21	1690.75	207.50	15
22	1090.88	258.13	22
23	1590.63	354.13	22
24	410.13	172.88	1
25	47.13	94.50	15
26	1752.50	151.13	36
27	192.00	364.25	28
28	576.63	128.63	18
29	1064.13	213.63	1
30	1119.88	482.75	34
31	1290.13	47.50	37
32	346.38	536.00	7
33	143.50	530.88	31
34	1706.38	226.00	21
35	796.38	319.63	17
36	759.38	463.38	11
37	1373.50	171.63	3
38	1639.00	596.75	36
39	356.75	499.25	3
40	1111.63	350.63	25
41	1473.25	510.63	29
42	177.25	298.38	12
43	1627.38	230.25	6
44	107.75	334.00	20
45	1453.13	146.13	5
46	1693.88	137.13	36

Component ID	X	Y	Component type
47	1504.63	553.50	37
48	254.25	523.00	23
49	992.00	316.13	21
50	1419.13	398.25	23
51	18.88	489.25	12
52	725.63	140.38	10
53	138.13	15.13	10
54	614.00	155.25	7
55	1391.13	433.13	1
56	604.25	531.13	40
57	462.00	123.75	11
58	419.38	508.38	12
59	586.25	111.50	13
60	885.13	151.13	9
61	452.75	257.63	32
62	1049.50	352.50	8
63	1196.13	145.38	20
64	193.38	43.38	14
65	1764.50	113.13	19
66	1646.63	551.75	4
67	991.13	341.50	22
68	600.75	104.25	4
69	1494.00	564.50	24
70	411.25	258.75	37
71	447.13	328.38	33
72	523.50	270.00	40
73	1341.75	462.63	19
74	1302.75	75.25	20
75	472.63	23.13	35
76	1508.63	101.50	28
77	1745.63	24.38	13
78	126.50	505.75	22
79	781.38	571.75	5
80	1061.50	166.00	3
81	1139.00	299.38	20
82	1207.88	104.25	1
83	450.38	60.50	20
84	721.75	55.88	21
85	109.75	380.00	11
86	1459.13	129.63	39
87	1658.13	562.38	17
88	1324.50	359.00	21
89	321.38	362.75	5
90	402.50	103.50	37
91	636.75	365.13	23
92	121.13	583.88	33
93	0.75	178.25	35
94	35.25	272.63	13
95	1527.75	171.75	1
96	932.50	486.00	6
97	1226.38	71.75	16

Component ID	X	Y	Component type
98	429.75	114.75	13
99	1466.75	398.63	29
100	1229.25	515.63	1
101	317.00	481.75	24
102	139.63	373.50	21
103	876.13	402.63	3
104	1547.63	446.38	39
105	982.63	67.25	10
106	140.25	515.88	25
107	272.63	370.88	4
108	1141.88	562.25	11
109	1382.63	551.25	13
110	1071.88	546.63	34
111	446.88	491.13	20
112	1134.38	143.88	3
113	600.63	243.63	33
114	50.88	379.63	3
115	1324.75	345.00	38
116	249.25	286.50	4
117	165.50	253.25	27
118	404.75	595.50	34
119	464.38	319.63	27
120	1433.00	373.75	10
121	2.50	148.00	14
122	1471.25	256.88	38
123	1079.00	315.50	39
124	378.88	501.50	15
125	1514.38	23.00	33
126	403.63	302.63	40
127	31.75	234.00	5
128	724.38	242.50	19
129	360.38	154.38	34
130	1735.13	460.38	1
131	288.25	399.88	31
132	1790.38	516.25	15
133	1731.00	200.13	24
134	740.50	529.50	33
135	705.63	388.38	14
136	21.00	525.75	40
137	869.50	583.88	21
138	117.38	318.63	2
139	434.75	219.38	4
140	1643.00	384.38	33
141	1.75	328.50	3
142	1498.13	545.25	12
143	902.38	283.00	2
144	1505.63	271.75	20
145	1241.25	515.13	18
146	602.00	43.63	34
147	465.38	584.88	22
148	536.63	367.25	31

Component ID	X	Y	Component type
149	355.13	167.38	3
150	1543.00	305.38	28
151	775.25	198.50	40
152	5.38	35.75	35
153	1490.13	66.25	16
154	371.75	444.75	3
155	371.25	458.13	5
156	781.38	562.25	9
157	574.38	337.63	2
158	635.25	346.63	30
159	106.63	557.13	20
160	742.75	207.13	29
161	481.25	317.50	35
162	1177.13	556.50	23
163	460.25	21.00	22
164	139.38	211.88	33
165	370.13	87.38	4
166	33.25	560.75	25
167	1608.63	549.88	32
168	1464.25	434.63	16
169	638.13	444.63	37
170	1524.13	194.00	23
171	1641.25	391.88	1
172	1284.13	544.50	27
173	1208.38	451.38	36
174	831.00	68.25	18
175	83.13	166.75	28
176	135.00	11.88	19
177	977.38	557.00	9
178	1623.50	359.38	9
179	1388.50	285.38	28
180	1188.50	385.00	24
181	1189.25	594.38	35
182	354.00	264.88	25
183	719.38	118.25	11
184	452.38	452.75	40
185	356.25	517.88	2
186	870.25	210.00	29
187	376.75	306.38	15
188	1354.00	433.13	32
189	640.88	596.00	18
190	52.63	166.13	23
191	106.00	301.38	27
192	6.50	20.75	16
193	428.88	92.75	7
194	1500.88	407.13	5
195	1516.25	169.63	8
196	468.63	412.63	21
197	767.88	307.50	3
198	389.88	208.63	7
199	471.50	397.75	26

Component ID	X	Y	Component type
200	485.00	373.75	28
201	431.50	242.25	2
202	124.25	561.75	5
203	450.88	307.63	40
204	430.13	379.38	18
205	300.88	554.25	36
206	320.38	302.63	6
207	1140.75	479.13	32
208	1290.75	80.63	21
209	1396.13	484.75	14
210	47.13	69.50	37
211	808.00	214.50	21
212	1414.13	66.13	38
213	54.13	216.63	10
214	574.38	388.75	6
215	559.63	347.00	5
216	485.50	557.00	1
217	404.13	440.25	26
218	773.63	149.25	5
219	664.00	459.75	10
220	66.88	163.88	3
221	1425.75	35.25	11
222	1379.13	543.75	1
223	722.13	88.88	8
224	1557.00	241.88	6
225	272.00	21.13	40
226	15.25	140.38	36
227	1477.00	514.63	13
228	624.25	191.75	4
229	1690.75	225.13	37
230	948.75	407.75	24
231	998.13	437.63	2
232	362.75	547.25	30
233	594.63	163.00	36
234	1343.13	245.25	30
235	1732.63	111.25	25
236	356.13	457.00	18
237	1259.75	226.38	8
238	222.00	316.25	25
239	1615.50	78.88	4
240	1110.13	545.25	11

Dataset N240K40_F: Feeder setup

Feeder slot	Component type
0	26
1	34
2	19
3	6
4	17
5	1
6	12
7	31
8	4
9	40
10	37
11	10
12	22
13	15
14	23
15	16
16	29
17	7
18	35
19	11
20	3
21	20
22	2
23	39
24	38
25	25
26	5
27	33
28	9
29	27
30	14
31	32
32	30
33	13
34	24
35	18
36	28
37	36
38	21
39	8

Dataset N80K20_D: PCB points

N=80, K=20

Component ID	X	Y	Component type
1	1505.13	584.63	16
2	1441.75	481.13	9
3	89.00	894.13	6
4	69.13	635.75	10
5	306.25	561.75	3
6	1965.00	999.00	8
7	976.88	28.25	1
8	815.13	524.88	2
9	923.00	534.75	17
10	1729.63	490.88	1
11	391.63	426.13	15
12	471.25	455.63	17
13	1142.63	141.75	16
14	280.13	355.63	2
15	618.50	771.38	6
16	842.75	797.38	12
17	1626.88	848.00	19
18	1478.00	402.13	8
19	304.63	792.50	13
20	1877.75	39.63	20
21	1215.25	873.25	17
22	1174.13	297.75	9
23	1752.25	347.63	14
24	1992.25	43.13	2
25	110.00	818.75	7
26	285.75	71.13	7
27	1690.25	269.13	15
28	1779.25	916.38	15
29	209.00	800.63	1
30	130.25	525.25	9
31	1379.50	247.75	14
32	128.38	550.50	3
33	649.00	116.63	4
34	1647.75	558.25	19
35	672.50	89.38	20
36	900.25	691.00	5
37	900.38	793.50	14
38	139.63	666.63	11
39	1748.50	344.00	14
40	1466.00	86.25	4
41	415.75	193.50	10
42	1848.75	960.00	16
43	1767.13	752.38	6
44	1113.50	787.75	12
45	1670.38	467.00	10
46	1190.13	941.13	1

Component ID	X	Y	Component type
47	537.38	416.38	17
48	39.88	54.38	2
49	647.25	292.13	11
50	549.50	781.75	10
51	1938.75	194.38	8
52	1493.50	561.63	5
53	1182.25	710.63	12
54	1053.63	623.63	18
55	323.88	558.50	9
56	263.38	963.63	14
57	507.38	941.25	7
58	1132.25	560.38	11
59	1585.25	678.38	16
60	529.75	739.88	4
61	1822.50	321.50	16
62	1972.63	535.38	9
63	722.38	640.25	5
64	1040.00	584.50	11
65	992.50	679.25	9
66	1555.38	823.25	1
67	69.38	260.38	18
68	1541.38	146.38	13
69	394.50	954.25	20
70	1497.25	815.00	8
71	349.50	926.38	8
72	1205.63	461.88	19
73	175.25	836.38	1
74	1439.75	161.13	5
75	619.13	680.63	2
76	1611.50	131.38	14
77	88.63	625.63	4
78	1438.13	316.63	10
79	1748.13	93.25	12
80	1502.38	416.13	12

Dataset N80K20_D: Feeder setup

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

Dataset N240K40_D: PCB points

N=240, K=40

Component ID	X	Y	Component type
1	748.38	33.88	15
2	90.50	139.50	28
3	340.13	692.38	25
4	1786.00	817.63	2
5	1363.00	379.88	30
6	400.75	638.38	4
7	29.50	374.00	10
8	169.25	912.75	27
9	1492.38	392.50	8
10	48.88	48.75	12
11	1125.00	256.63	25
12	729.75	79.00	25
13	1136.63	495.50	2
14	87.75	866.13	6
15	726.75	331.13	1
16	989.38	490.50	23
17	110.00	354.88	23
18	695.50	967.63	37
19	1483.13	366.88	38
20	1508.75	657.50	13
21	1421.88	49.75	18
22	1056.13	149.25	34
23	649.38	738.13	18
24	1662.88	292.38	3
25	790.38	349.50	11
26	528.50	536.00	27
27	1329.25	469.13	29
28	1294.75	573.00	33
29	1753.75	994.50	1
30	1970.38	894.88	12
31	131.13	114.13	7
32	358.13	669.38	34
33	85.63	52.75	24
34	1043.63	532.63	32
35	1899.63	518.63	9
36	1926.75	108.75	10
37	973.13	874.63	7
38	797.88	157.25	3
39	1334.75	657.75	3
40	1427.13	662.50	1
41	1117.00	136.50	37
42	511.50	597.75	13
43	572.38	456.88	3
44	734.38	42.75	35
45	1202.75	217.75	32
46	1444.88	611.25	28

Component ID	X	Y	Component type
47	1840.75	682.63	3
48	445.13	475.88	20
49	904.63	896.50	14
50	613.88	193.63	33
51	586.88	924.88	33
52	1344.63	433.00	34
53	1615.50	872.00	1
54	1013.75	570.38	40
55	1419.63	766.50	11
56	1208.50	295.50	22
57	1486.75	59.25	9
58	1633.25	559.13	4
59	289.25	901.63	11
60	1426.38	389.75	40
61	1793.88	905.13	37
62	971.75	254.38	15
63	1678.88	983.50	7
64	1866.63	763.38	26
65	441.38	668.50	32
66	524.00	736.00	31
67	900.13	274.75	11
68	1232.00	448.00	36
69	971.25	224.25	9
70	386.00	214.13	23
71	1421.75	749.75	2
72	1755.50	531.88	22
73	898.50	858.75	14
74	105.25	958.13	21
75	45.25	798.88	38
76	540.88	251.25	3
77	1173.75	20.25	7
78	1345.88	475.88	21
79	1889.63	977.50	6
80	1246.63	257.00	32
81	330.88	772.75	37
82	1936.88	932.25	7
83	1979.00	185.88	16
84	409.00	668.13	38
85	1958.38	999.00	10
86	902.88	836.50	23
87	403.75	705.63	4
88	448.75	678.25	23
89	1918.13	921.63	21
90	938.50	381.50	27
91	1118.38	69.00	3
92	1847.00	713.00	39
93	1581.50	482.88	8
94	1233.38	461.13	11
95	824.50	695.75	19
96	12.63	534.63	31
97	1966.63	467.25	6

Component ID	X	Y	Component type
98	1689.75	540.25	22
99	1150.50	635.38	2
100	1440.13	903.88	32
101	1114.38	256.63	16
102	1068.50	932.88	2
103	1530.25	68.25	23
104	1815.38	524.25	39
105	898.63	485.75	4
106	664.50	188.00	26
107	676.13	176.50	29
108	1657.63	60.75	2
109	671.25	874.75	12
110	802.75	807.50	7
111	1991.88	274.38	7
112	1098.75	896.63	4
113	610.38	654.63	16
114	307.88	989.25	33
115	1621.13	741.88	7
116	180.88	111.63	19
117	704.88	201.00	36
118	1800.63	904.25	3
119	1414.00	387.88	21
120	1609.50	360.88	9
121	178.00	555.63	33
122	115.63	29.38	34
123	1652.13	76.38	14
124	507.25	605.00	18
125	856.88	500.50	32
126	1511.50	662.75	9
127	1818.50	273.13	25
128	731.13	494.00	24
129	598.63	773.75	28
130	317.88	129.25	8
131	1250.38	492.88	6
132	606.38	765.50	12
133	786.50	217.25	27
134	1291.25	930.75	9
135	1121.13	849.25	2
136	467.00	606.50	18
137	1757.75	763.00	6
138	1205.25	162.88	31
139	1712.88	534.63	4
140	1654.13	382.88	32
141	1525.50	203.25	2
142	175.38	286.88	26
143	250.25	664.75	28
144	1925.13	234.88	9
145	913.63	301.75	3
146	1211.63	732.63	33
147	1942.63	724.13	22
148	1504.38	933.88	17

Component ID	X	Y	Component type
149	1963.38	798.25	23
150	353.50	584.63	18
151	1928.25	72.38	29
152	649.25	291.25	2
153	1172.63	555.25	13
154	978.13	935.75	25
155	1150.75	568.75	23
156	1233.50	32.25	8
157	507.38	239.25	32
158	317.75	712.38	35
159	1799.00	441.50	38
160	404.25	165.88	16
161	602.38	69.75	3
162	1328.75	651.63	35
163	1151.13	937.63	25
164	1258.88	71.13	21
165	72.25	848.25	10
166	1126.88	489.00	3
167	73.13	822.13	21
168	526.38	794.63	20
169	938.38	274.88	37
170	1817.50	756.63	12
171	1277.50	172.00	14
172	509.38	441.50	37
173	895.25	711.50	12
174	427.00	108.25	37
175	92.88	575.13	19
176	1889.50	250.00	2
177	344.13	451.88	23
178	1940.25	131.63	35
179	367.75	600.75	12
180	327.13	452.63	33
181	1311.13	61.50	1
182	136.25	851.50	11
183	1757.50	24.38	7
184	1216.63	289.50	5
185	1904.75	434.13	24
186	622.50	265.13	11
187	840.00	466.63	25
188	336.50	215.13	7
189	1302.50	495.88	15
190	1882.13	523.00	17
191	903.50	970.38	11
192	1867.25	975.88	5
193	1579.75	899.38	16
194	1898.63	342.75	16
195	1619.63	355.38	26
196	1130.63	346.63	2
197	25.38	771.63	19
198	1393.38	205.50	31
199	1913.63	178.25	21

Component ID	X	Y	Component type
200	846.88	924.50	31
201	1885.63	409.00	9
202	1103.13	900.13	7
203	575.00	473.63	36
204	167.13	934.88	3
205	397.50	399.00	30
206	675.63	46.38	9
207	1213.75	475.13	34
208	979.63	235.13	1
209	1623.00	873.75	10
210	797.88	113.13	30
211	397.50	711.63	5
212	1886.88	267.88	10
213	1132.38	232.50	7
214	1191.25	453.25	36
215	956.63	216.25	14
216	969.63	926.38	25
217	220.25	372.63	32
218	60.38	348.38	19
219	601.13	317.13	25
220	1031.38	189.50	19
221	1771.25	510.50	40
222	224.13	602.00	38
223	1489.38	393.25	5
224	1.25	72.88	9
225	1033.50	203.88	21
226	1617.88	929.38	5
227	1369.50	660.00	28
228	667.38	18.38	32
229	964.63	943.13	10
230	1990.88	635.00	22
231	731.00	48.38	22
232	464.88	248.38	31
233	1246.50	493.25	30
234	325.63	456.88	27
235	341.63	236.38	3
236	160.25	103.63	7
237	1922.88	170.75	9
238	319.50	267.50	36
239	948.38	225.75	22
240	530.00	505.25	9

Dataset N240K40_D: Feeder setup

Feeder slot	Component type
0	29
1	37
2	6
3	40
4	35
5	38
6	10
7	13
8	8
9	5
10	25
11	1
12	33
13	12
14	31
15	34
16	17
17	9
18	39
19	3
20	15
21	14
22	7
23	16
24	26
25	27
26	36
27	28
28	22
29	2
30	19
31	24
32	4
33	30
34	23
35	32
36	11
37	21
38	18
39	20

APPENDIX B: Dataset B

Dataset B: FEEDER table

FEEDER SETUP						
Component type	**Package name**	**Feeder slot**	**Feeder bank**	**X**	**Y**	**Feeder type**
10	A	1	A	-282.50	-170.00	reel
1	B	4	A	-237.50	-170.00	reel
2	C	7	A	-192.50	-170.00	reel
3	D	10	A	-147.50	-170.00	reel
6	I	12	A	-122.50	-170.00	reel
4	F	13	A	-102.50	-170.00	reel
5	G	15	A	-77.50	-170.00	reel
8	L	101	B	-282.50	170.00	reel
9	M	104	B	-237.50	170.00	reel
8	K	105	B	-212.50	170.00	reel
6	O	107	B	-192.50	170.00	reel
7	J	110	B	-147.50	170.00	reel
10	N	113	B	-102.50	170.00	reel
4	E	115	B	-77.50	170.00	reel

Dataset B: NOZZLE table

Position	Nozzle ID
1	1
2	1
3	2
4	2
5	4
6	8
7	16
8	32
9	64

Note:

Position - the location of the nozzle in the tool bank

Dataset B: PACKAGES table

Package ID	Package name	Component type	Component recognition	Nozzles
1	A	10	9	6
2	B	1	8	1
3	C	2	8	2
4	D	3	2	12
5	E	4	9	24
6	F	4	2	8
7	G	5	1	3
9	I	6	2	8
10	J	7	2	4
11	K	8	1	32
12	L	8	8	64
13	M	9	11	72
14	N	10	3	64
15	O	6	2	16

Note:

Component recognition - 1:SCC only; 2:LCC only; 3:SCC or LCC; 8: Mechanical only; 9: Mechanical or SCC, 10:Mechanical or LCC, 11: Mechanical, SCC or LCC where SCC, LCC and Mechanical are the component recognition/alignment methods that are using small vision camera, large vision camera and mechanical alignment on the fly, respectively .

Nozzles - is a combination of Nozzle ID i.e. the nozzles that can be used for pick-and-place the component package. The following are the examples of the Nozzles coding (using binary coding):

	T_8	T_7	T_6	T_5	T_4	T_3	T_2	T_1	T_0
Nozzles = 4	0	0	0	0	0	0	1	0	0
Nozzles = 6	0	0	0	0	0	0	1	1	0
Nozzles = 19	0	0	0	0	1	0	0	1	1

If Nozzles=4, the nozzle ID=4 can be used for the component package.
If Nozzles=6, the nozzle ID=4 or 2 can be used for the component package.
If Nozzles=19, the nozzle ID=16, 2 or 1 can be used for the component package.

Dataset B: PCBpoints table

Component ID	Component type	X	Y	Status
1	2	32.00	11.00	0
10	1	35.50	29.00	0
11	1	47.63	27.38	0
12	10	15.40	50.60	0
13	7	40.00	20.00	0
14	2	74.60	30.80	0
15	9	50.00	40.40	0
16	6	60.20	70.10	0
17	8	70.80	80.30	0
18	10	30.00	21.00	0
19	10	50.80	36.20	0
2	3	30.50	16.88	0
20	2	90.10	20.80	0
21	5	110.60	60.50	0
22	10	84.00	60.60	0
23	7	48.00	74.80	0
24	3	38.00	92.20	0
25	9	24.60	49.60	0
26	8	28.20	95.70	0
27	4	58.80	110.60	0
28	9	84.80	102.10	0
29	6	56.20	83.90	0
3	3	26.00	26.50	0
30	5	38.60	94.80	0
4	5	71.00	19.00	0
5	4	32.38	35.00	0
6	5	78.88	26.25	0
7	1	35.50	45.75	0
8	2	52.25	37.00	0
9	4	54.88	25.75	0

Note:

Status - *the status of PCB point i.e. 0: unschedule (available to be schedule), 1: scheduled but not placed yet, 2: being placed, 3:placed, 4:unavailable.*

APPENDIX C: Dataset DIMA

Dataset DIMA: FEEDER table

FEEDER SETUP						
Component type	**Package name**	**Feeder slot**	**Feeder bank**	**X**	**Y**	**Feeder type**
100NF-1206	RC1206X	34	1	-42.50	-170	reel
100NF-TNT1	TNT-D	23	1	-207.50	-170	reel
10K-1206	RC1206	18	1	-282.50	-170	reel
10NF-MELF	MELF	24	1	-192.50	-170	reel
1K-0402	RC0402	19	1	-267.50	-170	reel
1NF-MINIMELF	MELF-MINI	25	1	-177.50	-170	reel
2K2-0603	RC0603	20	1	-252.50	-170	reel
4K7-0805	RC0805	21	1	-237.50	-170	reel
AMD-253	SOL28	35	1	-23.45	-170	stick
BC-547	SOT23	22	1	-222.50	-170	reel
BLY-98	SOT143	33	1	-57.50	-170	reel
DIPSW	DIL-8	31	1	-87.50	-170	reel
FM-256k-Mc	TSOP32	50	2	-177.50	170	reel
HCT-273	SO16	41	2	-237.50	170	stick
HEF-2453	SOL24	41	2	-252.50	170	stick
KLM-202	PLCC44	56	2	-87.50	170	reel
KLPD-101	PLCC28	41	2	-267.50	170	stick
KLU-343	PLCC84	38	1	21.55	-170	reel
MKR-102	LQFP-120	48	2	-192.50	170	reel
MRF-337	PLCC68	53	2	-132.50	170	reel
OPM-479	SO8	39	2	-282.50	170	reel
PE-161	QFP-208	46	2	-207.50	170	reel
RAM-32K	SOJ28	27	1	-138.89	-170	stick
SAB-8089-2b	QFP-100	43	2	-222.50	170	reel
T-BCX54	SOT89	26	1	-162.50	-170	reel
TO252AA	TO252AA=DPACK	27	1	-111.59	-170	stick

Dataset DIMA: NOZZLE table

Position	Nozzle ID
1	1
2	2
3	4
4	8
5	16
6	32
7	64
8	128
9	256

Dataset DIMA: PACKAGES table

ID	Package name	Component type	Recognition	Nozzles
2	DIL-8	DIPSW	1	96
7	MELF	10NF-MELF	1	4
9	MELF-MINI	1NF-MINIMELF	1	2
12	PLCC28	KLPD-101	1	224
14	PLCC44	KLM-202	1	448
16	PLCC68	MRF-337	1	384
17	PLCC84	KLU-343	2	384
19	QFP-100	SAB-8089-2b	1	384
24	QFP-208	PE-161	2	384
30	RC0402	1K-0402	1	1
31	RC0603	2K2-0603	1	3
32	RC0805	4K7-0805	1	2
33	RC1206	10K-1206	1	6
33	RC1206X	100NF-1206	1	6
48	SOL28	AMD-253	1	224
51	SO8	OPM-479	1	24
55	SOJ28	RAM-32K	1	224
59	SOT23	BC-547	1	3
71	TSOP32	FM-256k-Mc	1	224
75	LQFP-120	MKR-102	1	448
290	SO16	HCT-273	10	8
323	TO252AA=DPACK	TO252AA	2	112
378	SOL24	HEF-2453	1	32
380	SOT143	BLY-98	1	6
389	SOT89	T-BCX54	1	12
395	TNT-D	100NF-TNT1	1	24

Dataset DIMA: PCBpoints table

Component ID	Component Type	X	Y	Status
R76	10K-1206	61.98	56.01	0
R84	10K-1206	23.88	50.29	0
R83	10K-1206	20.07	50.29	0
R82	10K-1206	16.26	50.29	0
R81	10K-1206	12.45	50.29	0
R80	10K-1206	8.64	50.29	0
R8	4K7-0805	20.47	8.19	0
R79	10K-1206	4.83	50.29	0
R69	10K-1206	35.31	56.01	0
R77	10K-1206	65.79	56.01	0
R87	10K-1206	35.31	50.29	0
R75	10K-1206	58.17	56.01	0
R74	10K-1206	54.36	56.01	0
R73	10K-1206	50.55	56.01	0
R72	10K-1206	46.74	56.01	0
R71	10K-1206	42.93	56.01	0
R70	10K-1206	39.12	56.01	0
T4	BC-547	87.31	18.14	0
R78	10K-1206	69.60	56.01	0
R94	10K-1206	61.98	50.29	0
R33	4K7-0805	121.82	20.47	0
T2	BC-547	93.07	23.18	0
T1	BC-547	87.31	23.14	0
SW1	DIPSW	72.39	39.49	0
R99	1K-0402	50.72	81.79	0
R98	1K-0402	50.72	83.06	0
R97	1K-0402	50.72	84.33	0
R85	10K-1206	27.69	50.29	0
R95	10K-1206	65.79	50.29	0
R86	10K-1206	31.50	50.29	0
R93	10K-1206	58.17	50.29	0
R92	10K-1206	54.36	50.29	0
R91	10K-1206	50.55	50.29	0
R90	10K-1206	46.74	50.29	0
R9	4K7-0805	23.92	7.28	0
R89	10K-1206	42.93	50.29	0
R88	10K-1206	39.12	50.29	0
R68	10K-1206	31.50	56.01	0
R96	10K-1206	69.60	50.29	0
R40	4K7-0805	102.51	6.94	0
R49	2K2-0603	105.21	12.19	0
R48	2K2-0603	107.83	12.89	0
R47	2K2-0603	110.28	14.02	0
R46	2K2-0603	112.51	15.59	0
R45	2K2-0603	114.42	17.50	0
R44	2K2-0603	115.99	19.72	0
R43	2K2-0603	117.11	22.17	0
R7	4K7-0805	17.22	9.70	0

Component ID	Component Type	X	Y	Status
R41	2K2-0603	118.06	27.50	0
R51	1K-0402	113.05	27.50	0
R4	4K7-0805	9.70	17.22	0
R39	4K7-0805	106.09	7.28	0
R38	4K7-0805	109.54	8.19	0
R37	4K7-0805	112.79	9.70	0
R36	4K7-0805	115.70	11.75	0
R35	4K7-0805	118.26	14.30	0
R34	4K7-0805	120.31	17.22	0
R42	2K2-0603	117.81	24.80	0
R59	1K-0402	104.34	17.11	0
R67	10K-1206	27.69	56.01	0
R66	10K-1206	23.88	56.01	0
R65	10K-1206	20.07	56.01	0
R64	10K-1206	16.26	56.01	0
R63	10K-1206	12.45	56.01	0
R62	10K-1206	8.64	56.01	0
R61	10K-1206	4.83	56.01	0
R5	4K7-0805	11.75	14.30	0
R6	4K7-0805	14.30	11.75	0
R50	2K2-0603	102.51	11.95	0
R58	1K-0402	106.13	17.58	0
R57	1K-0402	107.78	18.36	0
R56	1K-0402	109.30	19.41	0
R55	1K-0402	110.59	20.71	0
R54	1K-0402	111.64	22.23	0
R53	1K-0402	112.43	23.88	0
R52	1K-0402	112.89	25.66	0
T5	T-BCX54	78.97	15.97	0
R60	1K-0402	102.51	16.95	0
R120	2K2-0603	56.14	67.82	0
R13	2K2-0603	12.89	22.17	0
R128	4K7-0805	61.96	67.82	0
R127	4K7-0805	61.96	70.36	0
R126	4K7-0805	61.96	72.90	0
R125	4K7-0805	61.96	75.44	0
R124	4K7-0805	61.96	77.98	0
R123	4K7-0805	61.96	80.52	0
R113	2K2-0603	56.14	81.15	0
R121	4K7-0805	61.96	85.60	0
R16	2K2-0603	17.50	15.59	0
R12	2K2-0603	12.19	24.80	0
R119	2K2-0603	56.14	69.72	0
R118	2K2-0603	56.14	71.63	0
R117	2K2-0603	56.14	73.53	0
R116	2K2-0603	56.14	75.44	0
R115	2K2-0603	56.14	77.34	0
T3	BC-547	92.99	18.10	0
R122	4K7-0805	61.96	83.06	0
R23	1K-0402	17.58	23.88	0

Component ID	Component Type	X	Y	Status
R31	4K7-0805	123.06	27.50	0
R30	1K-0402	27.50	16.95	0
R3	4K7-0805	8.19	20.47	0
R29	1K-0402	25.66	17.11	0
R28	1K-0402	23.88	17.58	0
R27	1K-0402	22.23	18.36	0
R26	1K-0402	20.71	19.41	0
R14	2K2-0603	14.02	19.72	0
R24	1K-0402	18.36	22.23	0
R15	2K2-0603	15.59	17.50	0
R22	1K-0402	17.11	25.66	0
R21	1K-0402	16.95	27.50	0
R20	2K2-0603	27.50	11.95	0
R2	4K7-0805	7.28	23.92	0
R19	2K2-0603	24.80	12.19	0
R18	2K2-0603	22.17	12.89	0
R17	2K2-0603	19.72	14.02	0
R112	2K2-0603	56.14	83.06	0
R25	1K-0402	19.41	20.71	0
U2	RAM-32K	109.22	99.38	0
C2	100NF-TNT1	10.80	38.42	0
C1	100NF-TNT1	5.72	38.42	0
U9	MKR-102	102.33	42.28	0
U8	OPM-479	84.46	53.55	0
U7	SAB-8089-2b	104.70	67.21	0
U6	HCT-273	81.92	63.08	0
U5	HEF-2453	79.38	74.40	0
R114	2K2-0603	56.14	79.25	0
U3	FM-256k-Mc	107.29	86.42	0
C5	1NF-MINIMELF	24.76	38.42	0
U13	KLU-343	48.01	29.84	0
U12	MRF-337	48.01	29.84	0
U11	KLM-202	48.01	29.84	0
U10	KLPD-101	48.01	29.84	0
U1	PE-161	29.26	83.63	0
T7	TO252AA	76.91	23.05	0
T6	BLY-98	70.61	15.62	0
U4	AMD-253	78.11	87.74	0
R103	1K-0402	50.72	76.71	0
R111	2K2-0603	56.14	84.96	0
R110	1K-0402	50.72	67.82	0
R11	2K2-0603	11.95	27.50	0
R109	1K-0402	50.72	69.09	0
R108	1K-0402	50.72	70.36	0
R107	1K-0402	50.72	71.63	0
R106	1K-0402	50.72	72.90	0
C3	10NF-MELF	15.24	38.42	0
R104	1K-0402	50.72	75.44	0
C4	10NF-MELF	20.32	38.42	0

Component ID	Component Type	X	Y	Status
R102	1K-0402	50.72	77.98	0
R101	1K-0402	50.72	79.25	0
R100	1K-0402	50.72	80.52	0
R10	4K7-0805	27.50	6.94	0
R1	4K7-0805	6.94	27.50	0
C7	100NF-1206	108.88	99.38	0
C6	1NF-MINIMELF	28.58	38.42	0
R32	4K7-0805	122.73	23.92	0
R105	1K-0402	50.72	74.17	0