AN ABSTRACT OF THE DISSERTATION OF

Cumhur Alper Geloğulları for the degree of Doctor of Philosophy in Industrial Engineering presented on September 22, 2005.

Title: Group–Scheduling Problems in Electronics Manufacturing

Abstract approved: Redacted for Privacy

This dissertation addresses the "multi–machine carryover sequence–dependent group–scheduling problem with anticipatory setups," which arises in the printed circuit board (PCB) manufacturing. Typically, in PCB manufacturing different board types requiring similar components are grouped together to reduce setup times and increase throughput. The challenge is to determine the sequence of board groups as well as the sequence of individual board types within each group. The two separate objectives considered are minimizing the makespan and minimizing the mean flow time.

In order to quickly solve the problem with each of the two objectives, highly effective metasearch heuristic algorithms based on the concept known as *tabu search* are developed. Advanced features of tabu search, such as the long–term memory function in order to intensify/diversify the search and variable tabu–list sizes, are utilized in the proposed heuristics.

In the absence of knowing the actual optimal solutions, another important challenge is to assess the quality of the solutions identified by the proposed meta-heuristics. For that purpose, methods that identify strong lower bounds both on the optimal makespan and the optimal mean flow time are proposed. The

quality of a heuristic solution is then quantified as its percentage deviation from the lower bound. Based on the minimum possible setup times, this dissertation develops a lower bounding procedure, called procedure Minsetup, that is capable of identifying tight lower bounds.

Even tighter lower bounds are identified using a mathematical programming decomposition approach. Novel mathematical programming formulations are developed and a branch-and-price (B&P) algorithm is proposed and implemented. A Dantzig–Wolfe reformulation of the problem that enables applying a column generation algorithm to solve the linear programming relaxation of the master problem is presented. Single–machine subproblems are designed to identify new columns if and when necessary. To enhance the efficiency of the algorithm, approximation algorithms are developed to solve the subproblems. Effective branching rules partition the solution space of the problem at a node where the solution is fractional. In order to alleviate the slow convergence of the column generation process at each node, a stabilizing technique is developed. Finally, several implementation issues such as constructing a feasible initial master problem, column management, and search strategy, are addressed.

The results of a carefully designed computational experiment for both low–mix high–volume and high–mix low–volume production environments confirm the high performance of tabu search algorithms in identifying extremely good quality solutions with respect to the proposed lower bounds.

Group–Scheduling Problems in Electronics Manufacturing

by

Cumhur Alper Geloğulları

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented September 22, 2005
Commencement June 2006

Doctor of Philosophy dissertation of Cumhur Alper Geloğulları presented on September 22, 2005

APPROVED:

Redacted for Privacy

Major Professor, representing Industrial Engineering

Redacted for Privacy

Head of the Department of Industrial and Manufacturing Engineering

Redacted for Privacy

Dean of the Graduate School

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

Redacted for privacy

Cumhur Alper Geloğulları, Author

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to my major professor, Dr. Rasaratnam Logendran, for his support and guidance over the course of the last four years. His strong enthusiasm for research has been an invaluable source of inspiration for me. I also appreciate his help with my future career.

My gratitude extends to my committee members Dr. Jeffrey L. Arthur, Dr. J. David Porter, and Dr. Michael J. Quinn, for their constant encouragement and useful feedback, which have been of greatest help at all times. Special thanks to Dr. Porter and Dr. Arthur for serving as my minor professors, and for their help with my future career. I would like to thank Dr. Çetin Koç for serving as the graduate council representative in my committee for a while. I would like to express my gratitude to Dr. John Sessions for taking over this position in my committee.

I would like to thank the National Science Foundation for the financial support (NSF Grant No. DMI–0010118) while I was a research assistant during the pursuit of this research.

I would like to extend my appreciation to the IME staff members Denise Emery, Jean Robinson, and Phyllis Helvie for their helps. Special thanks to Bill Layton who installed and maintained any software/hardware that I needed for my research.

A number of people made the last four years bearable with their keen friendship. I wish to thank Tufan and Kerem for the long Mocha breaks at Starbucks and Dutch Bros., rakı nights, and late night feasts at the Shari's; special thanks to Kerem for always saying yes to sushi. I would also like to thank

Nasser, Hadi, Gökay, and Deniz for their friendship. I thank my close friends Filiz Gürtuna, Özlem Akpınar, Çağrı Gürbüz, Ömer Güneş, and Özge Yılmaz for their constant support and long phone conversations we had whenever I needed.

I will always be grateful to Ersin Gündoğdu for constantly encouraging and supporting me at my desperate times. Thanks Ersin for always being there for me.

Thank you Funda for your understanding and patience, and for relieving me of my worries with your presence.

Last, but certainly not least, thanks Gülin for always cheering me up with your talent to look at things on the bright side; thanks mom and dad for your everlasting love and care. *Beni en iyi şekilde yetiştiren, sevgi ve destekleriyle hep yanımda olan anneme, babama, ve kardeşime sonsuz teşekkürler.*

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

Page

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF TABLES

LIST OF TABLES (Continued)

Table Page

LIST OF TABLES (Continued)

LIST OF APPENDICES

LIST OF APPENDIX FIGURES

LIST OF APPENDIX TABLES

LIST OF APPENDIX TABLES (Continued)

LIST OF APPENDIX TABLES (Continued)

DEDICATION

To my parents Hamit and Gülseren Geloğulları

and to my sister Gülin Geloğulları

Group–Scheduling Problems in Electronics Manufacturing

1. INTRODUCTION

The miniaturization of electronic components has been one of the most significant developments of the last two decades. Improved technologies and constant reduction in prices have led integrated circuits to appear in all walks of life. One of the cornerstones in this process is the ability of assembling large–scale printed circuit boards (PCB) in an economical way. The electronics industry today is a well–developed worldwide industry. Fierce global competition and rapid technological advancements result in increasing product variety and complexity, shrinking product life cycles, and decreasing profit margins. Consequently, electronics manufacturing companies –even the big names– have to adapt to rapid changes. This dynamic nature creates production planning and control problems that are hard, yet need to be solved quickly. However, it is only recently that the operational aspects of electronics manufacturing industry have been addressed and attempts have been made to apply operations research techniques to these problems.

A PCB is a laminated board that is assembled with several to thousands of electronic components of different kinds, that is, with different sizes, shapes, and with different functions. A PCB consists of one or more layers of metal conductor and insulating material that allow electronic components to be interconnected and mechanically supported. The simplest form of a PCB is the single–layer single–sided board, which contains metal conductor on one side of the board only. On the other hand, multi–layer and double–sided boards provide greater levels of complexity and component density. In double–sided PCB assembly the compo-

nents are assembled on both sides of the board, and multilayering permits tracks to cross over one another, giving the designer more freedom in component layout [70]. Electronic components are either inserted through holes in the copper tracks on the boards and soldered in position, or are placed directly on the surface of the board and soldered. As a result, two distinctly different PCB assembly technologies have emerged. The conventional method is known as *through hole plated assembly* and is still popular especially for low–volume and manual assembly. Modern *surface mount technology* (SMT) utilizes smaller flat components which are well suited to automated assembly process. They are common in small consumer products such as cellular phones, whereas through hole components are still widely used in larger products like televisions and computer monitors where the competitive product price is a key factor.

PCBs are, by far, the most common embodiment of the electronic products. Over the years, the technologies of electronic components, as well as the technologies of PCBs themselves have changed dramatically, driven by the desire for greater functionality, reliability, and flexibility in smaller products. Consequently, PCB assembly has evolved from a labor–intensive activity to a highly automated one, characterized by steady innovations at the levels of design and manufacturing processes. PCB manufacturing became a capital–intensive activity involving technologically advanced complex processes using highly expensive equipment. Manual assembly methods may provide some flexibility but they cannot ensure a fast and reliable placement of the components on the PCBs. However, consumers expect reliable and cheap products, and thus, a common goal in PCB assembly is to put more functions into a board with the same size and cost [116]. Therefore, precise component placement is a necessity underlined by the need to use components and PCBs of ever decreasing sizes and tolerances. As a result, a vast

majority of the PCB assembly processes require *automated placement machines* that place (insert) the components on the boards. These machines allow faster and reliable assembly. However, the introduction of such sophisticated equipment has exacerbated an already difficult challenge of PCB production planning and control. The demand for high–speed and precise operation and flexibility in tooling makes it a very difficult task to control operation on these machines, while the competition faced by PCB manufacturers results in the need for high levels of utilization and efficiency, which can be achieved through the optimization of the production processes. All these features, combined with high initial investments and operating costs, and numerous constraints and managerial objectives, pose a challenge to production planning and control activity in PCB manufacturing.

PCBs are characterized by designs that range from simple low–value boards to very complex high–value ones. One electronics manufacturer may assemble PCBs with frequent design changes in small quantities (high–mix low–volume production), whereas another may assemble a large number of PCBs with designs that are fixed for a longer period of time (low–mix high–volume production). This results in the need to address the PCB assembly processes for both types of manufacturers. A recent development in PCB assembly is the growing role of *contract manufacturing* [91]. Contract manufacturers build a variety of products for many different customers. Original equipment manufacturers, on the other hand, build only their own products. Many original equipment manufacturers have abandoned their own assembly lines in favor of outsourcing the manufacturing functions to contract manufacturers. Despite the higher product variety and more dynamic product demand, contract manufacturers are expected to operate more efficiently than the original equipment manufacturers. This trend further emphasizes the importance of optimizing the PCB assembly processes.

Although this dissertation is not about the whole PCB assembly process itself, it is necessary to describe the process to better understand the problems we address. Before discussing the fundamental issues involved in the PCB production planning and control later in detail, we describe the *generic steps* involved in the assembly of a PCB. For our purpose, PCB assembly consists of placing a number of electronic components at prescribed locations on the bare board using automated placement machines. The placement machines are of various types. This is somewhat unfortunate from the operations researcher's point of view since the characteristics of the machines highly influence the nature of the production planning and control problems, and the formulation of the associated models. However, a *generic* description of an automated placement machine can be given as follows.

Each machine has a *worktable*, a *feeder carrier*, and a *pick–and–place device* as depicted in Figure 1.1. The bare boards are either placed by the operator or automatically transported onto the worktable that holds the board during the component placement process. Depending on the machine, the table can be stationary or mobile in the X–Y plane. The components that are packaged into component *tapes, sticks,* or *tubes* are loaded on the slots of the feeder carrier prior to production. Usually, the carrier can move along the X axis. The pick–and–place device retrieves the components from the feeder slots usually by vacuum, realigns them mechanically or optically, and places them at the appropriate locations on the board. Different designs and operating modes exist for the pick–and–place device. In one case, it moves in the Y–Z plane, picking up a component from the feeder and *then* placing it on the board. The pick–and–place device depicted in Figure 1.1 (also referred to as a rotary turret head), however, utilizes 8 workheads arranged circularly on a turret. Workhead 1 picks a component while workhead 5

simultaneously places another one on the board. Thereafter, the device turns 45° and a similar operation is repeated. The number of workheads on the machines may vary. Some placement machines are flexible in the sense that they can handle a wide range of board sizes as well as a wide range of different components, while others are restricted to a condensed set of components but they can operate at much higher speeds. For a detailed description of automated placement machines and different technologies associated with them, the reader is referred to Crama et al. [30], Francis et al. [42], and McGinnis et al. [93].

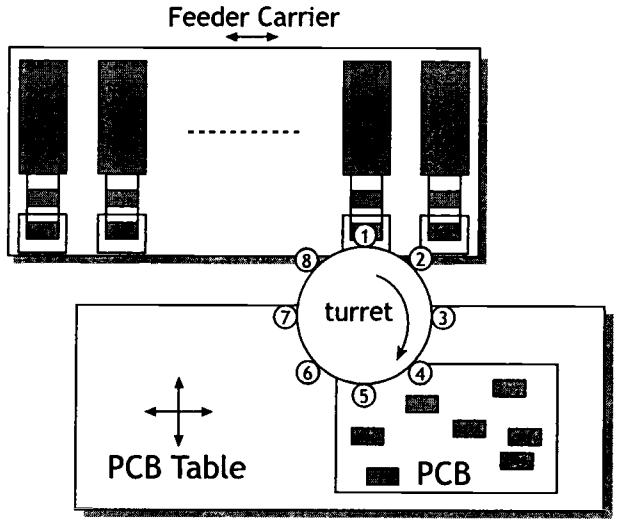

Figure 1.1. A Generic Automated Placement Machine

Most often in electronics manufacturing, the placement machines are laid out into distinct *assembly lines* arranged in a flowshop fashion. A flowshop is a configuration where the machines are set up in series and the jobs to be produced go through the machines sequentially. A flowshop is referred to as a *permutation* flowshop if the jobs are processed in the same order on all machines. Usually, a conveyor connects the machines within each line, transporting individual boards or batches of boards between the machines.

Production planning and control decisions are frequently addressed in a hierarchical framework where they decompose into long term (strategic), medium term (tactical), and short term (operational) issues. Strategic and tactical issues include decisions such as determining the shop floor layout, construction of machine and job groups, and assigning component feeders to machines. Operational decisions involve sequencing the component placement operations on the machines, scheduling of the PCB assembly, and similar shop floor operations.

Scheduling decisions in almost all types of manufacturing processes should be made rapidly since these decisions commonly involve the determination of the schedule for the next production period. Since electronic products become obsolete very quickly and as a result the manufacturers keep adding new PCBs to their product mix, production scheduling in PCB assembly often spans a period ranging from one shift to a day. The production schedule should be determined at least the night before so that it could be executed the next day. These schedules should serve in the best interest of the manufacturing company, i.e., they should optimize appropriate measures of performance.

The measure of performance most commonly considered in the literature is minimization of the *makespan.* The makespan, also referred to as the *total completion time* or the *maximum completion time* in the literature, is defined as the amount of time between the moment the first job enters the system and the moment the last job leaves the system. Typically, electronics manufacturing companies receive job orders from a variety of customers, each requiring PCBs of various types and quantities be produced and delivered at a desired due date. From time to time, a job order may be received from a customer for production of a large number of board types in fairly large quantities at the earliest possible time. In such instances, it would be beneficial for the electronics manufacturing

companies to consider minimizing the makespan of the orders, just so that all of the finished PCBs can be sent out in one shipment to reduce the transportation costs. An additional benefit might be when there is very limited area for storing finished goods inventory. In that case, batching customer requirements will help reduce finished goods inventory. On the other hand, if the job orders are received from different customers and if the due dates desired by them are not so stringent, the manufacturers should focus on minimizing the *mean flow time* of all the jobs considered for production. The mean flow time is defined as the average time the jobs spent in the production line, i.e., assuming that all the jobs are ready for production at time 0, it is the sum of the completion times of all the jobs divided by the total number of jobs. Consistent with manufacturing philosophies such as *just–in–time* and *lean manufacturing*, minimizing the mean flow time implicitly minimizes the mean work–in–process inventories, thus adheres to "world class manufacturing."

Group technology (GT) [61] is a popular methodology that has gained great interest from manufacturing companies all over the world. In GT, efficiencies are improved by grouping similar jobs according to shape, dimension, or process route. A significant advantage of applying GT principles to the manufacturing processes is that the setup times are reduced drastically. Since the jobs in a group are similar, saving of machine setup times is realized by processing jobs of the same group contiguously. Additional benefits of applying GT principles include reductions in cycle times, defective products, and machine tool expenditures, simplification of production scheduling and shop floor control, learning effects, and quality improvement [58]. In PCB assembly, the boards with similar components are typically included in one group, in which case the scheduling of PCBs falls under the category of *group–scheduling* and the issues concerning the assembly of

PCBs must be addressed at two levels. At the first level or the *individual board level* boards that belong to a group would have to be sequenced, and at the second level or the *board group level* the sequence of different board groups would have to be determined. Since the board types within the same group are similar, a change from a board type to another in the same group requires no setup time. However, as the groups themselves are dissimilar, the setup time required to switch from one board group to another is substantial that it cannot be disregarded. We shall discuss GT principles applied to PCB assembly, especially their role in the setup management strategies, in detail in the next chapter.

1.1. Research Contributions

This dissertation specifically addresses the group–scheduling of the PCB assembly processes in a multi–machine flowshop environment and is motivated by a real industry application. The focus is on two objectives, namely minimizing the makespan and minimizing the mean flow time. Most of the assembly lines in PCB production consists of two placement machines. Three–machine arrangements are also getting to be used lately. Hence, although our algorithms are applicable to assembly systems with an arbitrary number of machines, the primary interest is in two– and three–machine PCB assembly systems.

The problem investigated in this dissertation is shown to be $\mathcal{NP}$–hard in the strong sense with each of the two objectives, which means that it is highly unlikely that a polynomial time algorithm that optimally solves it exists. Already existing methods of operations research such as the branch–and–bound method would require an excessive amount of computation time to solve problems of in-

dustrial merit. Since the scheduling decisions must be made quickly, the interest is in developing fast algorithms that are able to identify high quality solutions.

For this reason, highly effective metasearch heuristic algorithms based on the concept known as *tabu search* are developed in order to quickly solve our scheduling problems. Advanced features of tabu search, such as the long–term memory function in order to intensify/diversify the search and variable tabu–list sizes, are utilized in the proposed heuristics. The tabu search based heuristics are designed to handle the two levels of the group–scheduling problem at two layers merged together, where each layer is a tabu search algorithm itself. Flexibility is a key factor in the actual production phase in electronics manufacturing since the same machinery is used to manufacture slightly different variants of the same product as well as a range of different product types. Apart from optimizing regular daily scheduling problems, the proposed tabu search heuristics provide a great deal of flexibility to the personnel in charge of scheduling the PCB assembly. The proposed tabu search based heuristics provide a basis for a flexible decision support system to easily cope with situations when a major change occurs, such as product mix changes that happen frequently in electronics manufacturing, urgent prototype series production requests, temporary lack of the required components, and machine breakdowns.

In the absence of knowing the actual optimal solutions, another challenge is to assess the quality of the solutions identified by the metasearch heuristics. For that purpose, methods that identify strong lower bounds both on the optimal makespan and the optimal mean flow time are proposed. The quality of a solution is then *quantified* as its percentage deviation from the lower bound. Based on the minimum possible setup times, we propose a lower bounding procedure for the

two–machine problem, called procedure *Minsetup*, which is capable of identifying tight lower bounds.

Even tighter lower bounds, for two– as well as three–machine problems, can be identified using a mathematical programming decomposition approach. Two novel modeling frameworks are proposed. These modelling frameworks can be used to formulate a mathematical programming model for any group–scheduling problem. Utilizing one of the mathematical formulations, a novel *branch–and–price* (B&P) algorithm is proposed for the group–scheduling problems addressed in this dissertation. Most of the column generation and B&P algorithms in the literature proposed for machine scheduling address parallel–machine problems. The proposed B&P algorithm, on the other hand, is one of the few that are developed for the flowshop scheduling problems (with sequential machines) and, to the best of our knowledge, is the first B&P approach addressing group–scheduling problems.

A Dantzig–Wolfe reformulation of the problem is constructed and a column generation algorithm is described to solve the linear programming relaxation of the Dantzig–Wolfe master problem, where separate single–machine subproblems are designed to identify new columns if and when necessary. To enhance the efficiency of the algorithm, approximation algorithms are developed to solve the subproblems as well. Ways to obtain valid global lower bounds in each iteration of the column generation algorithm are shown, and effective branching rules are proposed to partition the solution space of the problem at a node where the solution is fractional. These branching rules are incorporated into into the subproblems in an efficient way, as well. In order to alleviate the slow convergence of the column generation process, a stabilizing method is developed. The method, based on a primal–dual relationship, bounds the dual variable values at

the beginning and gradually relaxes the bounds as the process marches towards optimal dual variable values. Finally, several implementation issues, such as constructing a feasible initial master problem, column management, search strategy, and customized termination criteria are addressed.

The results of a carefully designed computational experiment based on the data generated for both *low–mix high–volume* and *high–mix low–volume* production environments confirm the high performance of tabu search algorithms in identifying extremely good quality solutions with respect to the proposed lower bounds.

1.2. Outline of the Dissertation

Chapter 2 reviews the literature on group–scheduling and on the production planning and control problems encountered in electronics manufacturing, focusing primarily on the scheduling of the PCB assembly processes. The impact of group technology principles on the PCB assembly processes especially in machine setup management, is discussed.

In chapter 3, the problem is described in detail, a representative example from the industry is presented, and the $\mathcal{NP}$–hardness of the problem with each of the two measures of performance is proved.

Based on the two novel modelling frameworks that are proposed for formulating group–scheduling problems, chapter 4 presents three mathematical programming formulations of the problem. These formulations are compared in terms of the number of variables and their ability to solve problems of various sizes.

Chapter 5 provides a brief background on the tabu search concept, describes the components and algorithmic structures of the proposed tabu search

algorithms, and demonstrates one of the algorithms on the representative example problem.

Chapter 6 presents procedure *Minsetup* and develops algorithms that guarantee valid lower bounds on the optimal makespan and the optimal mean flow time. These algorithms are demonstrated on the representative example problem.

Chapter 7 proposes a branch–and–price algorithm. First, the necessary background is given and then master and subproblem formulations are presented, valid lower bounds are developed, alternative strategies to solve column generation subproblems are discussed. In addition, stabilization and acceleration technique, and an effective branching rule is described. The chapter further discusses important implementation issues relevant to branch–and–price such as constructing a feasible initial master problem, column management strategies, and customized termination criteria.

Chapter 8 presents a carefully designed computational experiment to test the performance of the proposed metaheuristics with respect to the proposed lower bounds, and performs statistical analysis to interpret the results.

Finally, chapter 9 concludes the dissertation with the discussion of the results and contributions, and introduces ideas for future research.

2. LITERATURE REVIEW

Production scheduling is concerned with the allocation of a limited number of machines with limited capabilities to jobs over time. It is a decision making process with the goal of optimizing one or more objectives. Since there is an enormous number of research efforts reported in the published literature on this subject, the focus here is on group–scheduling problems rather than the whole scheduling literature.

The next section reviews group–scheduling in traditional manufacturing. Then, the literature pertinent to the production planning and control problems encountered in electronics manufacturing is reviewed, primarily focusing on scheduling of the PCB assembly processes.

In the light of the previous research efforts in the published literature, the PCB assembly scheduling problem with "carryover" sequence–dependent setup times is identified as a highly relevant problem in the industry. However, as shall be discussed later in this chapter, all of the previous research efforts on scheduling of the PCB assembly processes, which draw attention to the fact that the setup time required of the next board type/group depends on all of the previous board types/groups and the order they were processed (i.e., the setup operation is *carried over*), simplify the problem by using myopic approaches in order to *estimate* the setup times. This certainly results in losing valuable information about the problem and leads to inferior solutions. The mathematical programming models and algorithms proposed in this dissertation, on the other hand, address the carryover setup times structure as it is, explicitly evaluating the setup times rather than estimating them (except for procedure Minsetup presented in chapter 6).

2.1. Group–Scheduling in Traditional Manufacturing

In the past decades, there have been some efforts in the area of group–scheduling in traditional manufacturing. Liaee and Emmons [77] review the literature on scheduling families of jobs on a single machine or parallel machines. They consider scheduling with and without the *group technology assumption.* With the group technology assumption the jobs in the same family/group must be scheduled contiguously and the number of setups is equal to the number of groups, while without this assumption the jobs in the same group need not be scheduled contiguously. Scheduling under the group technology assumption is referred to as *group–scheduling* and is performed at two levels. At the first level individual jobs within each job group are sequenced, whereas at the second level job groups themselves are sequenced in order to optimize some measure of performance. The published research in group–scheduling can be classified as addressing problems with sequence–independent setup times and problems with sequence–dependent setup times.

2.1.1. Group–Scheduling with Sequence–Independent Setup Times

Yoshida and Hitomi [122] were the first to investigate the two–machine group–scheduling problem with sequence–independent setup times with the objective of minimizing the makespan. Their algorithm focuses on an extension of Johnson's algorithm [69] originally developed for the two–machine flowshop problem without setup times. Sule [114] extends this to a case where the setup, run, and removal times are separated, while Proust et al. [99] present heuristic

algorithms for minimizing the makespan of a multi–machine flowshop scheduling problem with setup, run, and removal times separated.

Pan and Wu [97] propose a heuristic algorithm for minimizing the mean flow time of a single–machine group–scheduling problem with the additional constraint that no jobs are tardy. Baker [14] proposes heuristics for minimizing the maximum lateness of the problem of scheduling job families with due dates and sequence–independent setup times on a single machine. The author uses earliest due date rule to sequence individual jobs within job groups, and develops conditions in order to use to construct group sequences. For minimizing the maximum lateness of a single–machine scheduling problem with release dates, due dates, and family setup times, Schutten et al. [108] propose a branch–and–bound algorithm that is reported to solve instances of reasonable sizes to optimality. To construct complete schedules, they treat the sequence–independent setup times as "setup jobs" with specific release dates, due dates, and precedence relations. Liao and Chuang [78] propose branch–and–bound algorithms for a single facility group–scheduling problem with the objectives of minimizing the number of tardy jobs and minimizing the maximum tardiness. Gupta and Ho [55] consider the problem of minimizing the number of tardy jobs to be processed on a single machine with two job classes where the setup times are sequence–independent. They propose a heuristic and incorporate branch–and–bound search feature to their heuristic. Yang and Chern [121] address a two–machine flowshop group–scheduling problem where each group requires sequence–independent setup and removal time, and there is a transportation time in between the machines. With the objective of minimizing the makespan, the authors propose a heuristic that generalizes already existing methods proposed in the literature for flowshop problems. For minimizing the makespan of a multi–machine group–scheduling prob-

lem with sequence–independent setup times, Schaller [106] proposes a two–phase heuristic that uses a branch–and–bound search in the first phase to construct family sequences and an interchange heuristic in the second phase to construct job sequences within families. The author develops lower bounds on the makespan to use in the branch–and–bound method.

Logendran and Nudtasomboon [85] develop a heuristic for minimizing the makespan of a multi–machine group–scheduling problem at the first level only. Their heuristic allows jobs to skip some machines and relies on the fact that jobs with higher mean processing time over all machines should be given a higher priority in generating partial schedules that eventually lead to generating complete schedules for the problem. In addition, Radharamanan [100] and Al–Qattan [6] document heuristic algorithms for minimizing the makespan of a group–scheduling problem at the first level. Logendran and Sriskandarajah [87] propose a heuristic algorithm for minimizing the makespan of a two–machine group–scheduling problem when there is blocking in between the machines. In the absence of the optimal solutions, they provide worst–case performance of two heuristics, and compare the quality of the heuristic solutions with random solutions. They analyze two cases. In the first case, the run times on the second machine is greater than the run times on the first machine, whereas in the second case the opposite is true. For solving group–scheduling problems, Allison [5] reports the performance of combining single–pass and multi–pass heuristics designed for flowshop problems. Logendran et al. [84] develop combined heuristics for minimizing the makespan of the multi–machine bi–level group–scheduling problems when setup and run times are separated.

2.1.2. Group–Scheduling with Sequence–Dependent Setup Times

Only a few studies address group–scheduling problems with sequence–dependent setup times. Strusevic [113] proposes a linear–time heuristic algorithm with a worst case ratio of 5/4 for an open shop group–scheduling problem with sequence–dependent setup times where the objective is minimizing the makespan. Schaller et al. [107] also address the group–scheduling problem with sequence–dependent setup times. Their scheduling problem originated from a PCB assembly process. They draw upon existing research on scheduling of the cellular manufacturing systems; they use existing methods to identify a job sequence within each group. For sequencing the job groups, they modify existing algorithms originally designed for flowshop scheduling problems with sequence–dependent setup times. Eom et al. [40] propose a three–phase heuristic algorithm for a parallel–machine group–scheduling problem with sequence–dependent setup times to minimize the total weighted tardiness. In the first phase jobs are grouped together according to similar due dates and are sequenced according to earliest due date rule. In the second phase the job sequences within groups are improved using a tabu search heuristic. In the third phase jobs are allocated to machines using a threshold value and a look–ahead parameter.

2.2. Production Planning and Control in PCB Manufacturing

Production planning decisions are frequently addressed in a hierarchical framework where they are decomposed into a number of more easily manageable subproblems. The main reason for the decomposition is that the production planning problems are usually too complex to be solved globally. It is easier to solve each subproblem one at a time. Then, the solution to the global problem can

be obtained by solving the subproblems successively. Obviously, this solution is not likely to be globally optimal, even if all subproblems are solved to optimality. Nonetheless, this approach is a productive and popular way to tackle hard problems [30]. A typical hierarchical decomposition scheme of the general planning problem discerns the following decisions.

1. *Strategic level* or *long term planning* concerns the initial deployment and subsequent expansion of the production environments (e.g., the design and selection of the equipment and the products to be manufactured).

2. *Tactical level* or *medium term planning* determines the allocation of production capacity to various products so that external demands are satisfied (e.g., by solving batching and loading problems).

3. *Operational level* or *short term planning* coordinates the shop floor production activities (e.g., by solving line sequencing problems).

Maimon and Shtub [89] relate these decisions to electronics assembly: In the strategic level the planning focuses on determining the best set of production equipment for the operation (e.g., running a simulation on how much money to invest in new equipment and what kind of machines to purchase). These decisions are usually made on economical basis and revised over long operational periods. At the tactical level the decisions concern machine and line configurations, batch sizes, and work–in–process inventory levels. Finally, the operational level addresses day–to–day operation of the equipment (e.g., how to manufacture a product). This dissertation addresses the problem of scheduling the PCB assembly, which is related to the operational level, and have to be solved quickly on a frequent basis.

Crama et al. [30] consider the long term decisions to be made and concentrate on tactical and operational decisions. In particular, they assume that the demand mix and the shop layout are fixed. Under these conditions, they identify the following subproblems to be addressed in the PCB assembly processes.

SP1. An *assignment* of PCB types to product families and to machine groups.

SP2. An *allocation* of component feeders to machines.

SP3. For each PCB type, a *partition* of the set of components, indicating which components are going to be placed by each machine.

SP4. For each machine group, a *sequence* of the PCB types, indicating in which order the board types should be produced on the machines.

SP5. For each machine, the *location of feeders* on the carrier.

SP6. For each pair of a machine and a PCB type, a *component placement sequence*, that is, a sequence of the placement operations to be performed by the machine on the PCB type.

SP7. For each pair of a machine and a PCB type, a *component retrieval plan*, that is, for each component on the board, a rule indicating from which feeder this component should be retrieved.

SP8. For each pair of a machine and a PCB type, a *motion control specification*, that is, for each component, a specification of where pick–and–place device should be located when it picks or places the component.

The subproblem SP1 is posed at the level of the whole assembly shop and involves all of the PCBs to be produced, SP2–SP4 usually arise for each PCB

family at the level of assembly lines or cells, and SP5–SP8 deal with individual machines. Usually, the subproblems SP6–SP8 are left to the placement machine manufacturer and the PCB manufacturer deals with the remaining subproblems. The PCB scheduling problem investigated in this dissertation falls under the subproblem SP4.

Other classification schemes have also been proposed. Häyrinen et al. [60] divide the PCB assembly problems into four general classes:

1. *Single-machine optimization* deals with optimizing the operation of a single machine. The problems such as feeder arrangement, placement sequencing, and component retrieval are addressed.

2. *Line balancing* involves balancing machine workload and eliminating bottlenecks.

3. *Grouping* involves identifying PCB groups and allocating the components to machines.

4. *Scheduling* deals with sequencing the PCBs to be produced on the machines.

This section reviews the literature according to the classification scheme proposed by Johnsson and Smed [70]. They classify the literature on the PCB assembly according to the number of board types and machines present in the problem. Accordingly, they identify four problem classes as follows.

1. *One PCB type and one machine* class comprises of single machine optimization problems, where the goal is to minimize the total component placement time.

2. *Multiple PCB types and one machine* class comprises of setup strategies for a single machine, where the goal is to minimize the setup time of the machine.

3. *One PCB type and multiple machines* class concentrates on component allocation to similar machines, where the usual objective is balancing the workload of the machines in the same line (usually by eliminating bottlenecks).

4. *Multiple PCB types and multiple machines* class usually concentrates on allocating jobs to lines which includes routing, lot sizing, workload balancing between lines, and line sequencing.

Among the single–machine problems are feeder arrangement, placement sequencing, and component retrieval problems. Since the type and design of the machine has a major impact when solving these subproblems, most of the work have been based on variants of pick–and–place and rotary turret machines. For example, Crama et al. [28] address the component retrieval problem for a Fuji CPII machine. The placement sequence of components on the board and the assignment of component types to feeder slots are given. The problem is then to decide from which feeder slot each component should retrieved. Naturally, this problem emerges only if at least one component type is duplicated in the feeders. The authors describe a CPM/PERT network model of the problem and present a dynamic programming algorithm for solving the problem. Ahmadi and Wurgaft [2] assume that the allocation of feeders to machines is given and allow for precedence relations among operations. They are interested in finding large subsets of PCBs for which the precedence relations form an acyclic digraph. They propose to determine and replicate the smallest number of operations (using multiple copies of the same feeder) so as to remove all cycles from the precedence graph. Ahmadi et al. [1] consider the feeder location problem on a machine, show that the problem is $\mathcal{NP}$–hard, and present an approximation algorithm with a

worst case ratio of 3/2. Crama et al. [29] solve the placement sequencing subproblem by an exchange heuristic in which the component retrieval problem is kept fixed over a number of successive iterations and reoptimized once in a while. Dikos et al. [36] develop a genetic algorithm for the feeder location problem with multiple board types under the assumption that placement sequences are known in advance. Sadiq et al. [103] present a method for arranging feeder slots in a surface mount technology machine in a high–mix low–volume environment. The slots are assigned using a heuristic rule and then the slot rearrangement process tries to locate the components so that they are adjacent in the insertion sequence. The goal is to assign the components required by an individual PCB type adjacent to each other in order to save feeder carrier movements. Bard et al. [16] model the placement sequence of a rotary turret machine as a traveling salesman problem and solve it with the nearest neighbor heuristic. Moreover, they formulate the feeder assignment and component retrieval problem as a quadratic integer program and solve it with a Lagrangean relaxation approach.

Problem class 2 given above involves with strategies to reduce the setup times required of various board groups on a single machine. Setup strategy has a significant impact on the efficiency. The approaches to reduce setup times in electronics manufacturing are usually categorized as (a) reducing the setup frequency by enlarging the lot sizes and (b) by applying group technology (GT). Leon and Peters [76] classify the setup management strategies proposed in the literature related to PCB assembly as follows.

1. *Unique setup strategy* considers one board at a time and specifies the component–feeder assignment and the placement sequence, which minimize the placement time. This is a common strategy when dealing with a single product and a single machine in a high–volume production environment.

2. *Group setup strategy* forms families of similar board types. A setup operation is performed only when switching production from one family to another.

3. *Minimum setup strategy* attempts to sequence boards and determine component–feeder assignments to minimize the total setup time.

4. *Partial setup strategy* is characterized by partial removal of components from the machine when switching production from one board type to the next.

In unique setup strategy, during the setup operation, *all* the components are removed from the feeders and replaced with the ones required by the next board. Most of the work on a single machine discussed above involve the unique setup strategy.

In group setup strategy, the feeder assignment is determined for a board group or family. Any board in the group can be produced without changing the component configuration on the machine. During the setup, *all* the components are removed from the feeders and the components required of the next board group are placed on the feeders. There are variations of this strategy, where a certain set of common components are left on the machine, while the rest of the components, which are called *residual* or *custom* components, are added or removed as required. Carmon et al. [23] describe a group setup method for a high–mix low–volume production environment where the boards are produced in two stages. First, common components are loaded and inserted on the boards of the whole group, and then the residual components are loaded and inserted. Hashiba and Chang [59] address a case with a single machine and multiple board types when the objective is to minimize the number of setups. They decompose the setup problem into three subproblems: grouping PCB types, sequencing the groups, and assigning components to machines. The authors apply heuristic algorithms to each

of the subproblems individually. Furthermore, they experiment with a simulated annealing method and observe that it identifies better solutions than the heuristic decomposition approach. Bhaskar and Narendran [13] apply graph theory for grouping PCBs. The PCBs are modeled as nodes and their similarities as weighted arcs between the nodes. After that, a maximum spanning tree is constructed for identifying the PCB groups. Maimon and Shtub [89] present a mathematical programming formulation and a heuristic method for grouping a set of PCBs to minimize the total setup time subject to machine capacity constraints. A user–defined parameter indicates whether multiple loading of PCBs and components is allowed.

Minimum setup strategy attempts to sequence the boards and determines feeder assignments to minimize the total component setup time. The idea is to perform only the feeder changes required to assemble the next PCB. In general, similar product types are produced in sequence so that little changeover time occurs. Lofgren and McGinnis [81] point out two key decisions: One must determine in what sequence the PCBs are to be processed and what component types should be staged on the machine for each PCB type. The authors present a sequencing algorithm in which the PCBs are first sequenced and after that a setup is determined for each PCB. The heuristic determines myopically the "best" set of components to add/remove whenever the existing setup is not sufficient. Barnea and Sipper [10] consider a case with a single machine and recognize two subproblems: sequence and mix. They present a mathematical programming model of the problem and use a heuristic approach based on the *keep tool needed soonest* (KTNS) rule introduced by Tang and Denardo [115]. In each iteration, the algorithm generates a new partial PCB sequence by using a sequencing algorithm that determines the next PCB to be added in the sequence, and a mix algorithm that

updates the component mix using the KTNS rule. Günther et al. [56] address a typical surface mount technology production line and apply a minimum setup strategy approach in which the PCBs are sequenced so that each PCB has the maximum number of components common with its predecessor. The authors discern three different subproblems: PCB sequencing, component setup, and feeder assignment. They solve each of the subproblems with heuristic algorithms. Dillon et al. [37] discuss minimizing the setup time by sequencing PCBs on a surface mount technology production line. The authors present four variants of a greedy heuristic that aims at maximizing the component commonality whenever the PCB type changes. This is realized by using a component commonality matrix from which board pairs with a high number of common components can be identified.

Partial setup strategy specifies that only a subset of the feeders on a machine are changed when switching from one PCB to the next. This strategy resides in between the unique setup strategy and the minimum setup strategy, since some feeders remain on the machines permanently while others are changed as required by the next PCB type to be produced. For example, for very large batch sizes the partial setup strategy may result in the unique setup strategy. It can result in minimum setup strategy, changing a few feeders for small batch sizes and short placement times. Therefore, partial setup strategy allows more flexibility. There are variants of this strategy. Balakrishnan and Vanderbeck [15] describe a problem of grouping PCBs and forming machine groups where the components are partitioned into two classes: *Temporary* feeders are loaded on the machines as needed and unloaded whenever a batch is completed, while *permanent* feeders remain unchanged. They develop a mathematical programming model to determine the temporary and permanent feeders and use column generation to solve the problem. Leon and Peters [75] consider the operation of component place-

ment equipment for the assembly of PCBs, develop a heuristic utilizing partial setup strategy, and compare its solutions with the corresponding solutions when unique, minimum, and group setups are employed. The unique setup strategy dominates when batch sizes are large, while the group setup strategy dominates the minimum setup strategy for especially high–mix low–volume production since it considers all the PCBs.

Several researchers addressed the component allocation problems in the case of multiple machines. Ben–Arieh and Dror [19] consider assigning components to two insertion machines so that all boards in a production plan can be produced and the output rate is maximized. They present a mathematical programming formulation and develop a heuristic algorithm to solve the problem. Askin et al. [9] discuss a surface mount technology plant with multiple identical machines. They assume that the total assembly times of board groups are almost equal. With the objective of minimizing the makespan, the authors present a four–stage approach for grouping the boards and allocating the components to the machines: First, the boards are grouped into production families. Next, for each family, the component types are allocated to the machines. After that, the families are divided into board groups with similar processing times. Finally, the board groups are scheduled. Brandeau and Billington [21] present two heuristic algorithms for a production line composed of an automated work phase and a manual phase: *stingy component* heuristic tries to avoid assigning less frequently used components to the automated work phase, while *greedy board* heuristic assigns a whole board to a single work phase instead of splitting it. Hillier and Brandeau [62] address the same problem and present a new mathematical programming model and an improved heuristic based on the branch–and–bound technique.

Scheduling of multiple PCB types on multiple machines is addressed by several researchers as well. Johri [71] considers sequencing batches of PCBs in a production environment which has been active for some time and new jobs are inserted to the existing production sequence. With the goal of minimizing the imbalance of the machine workloads, the author proposes a heuristic, which ensures that the due dates are not violated. If possible, the number of setups is minimized as well. First, the heuristic determines a desirable production rate for each workstation. After that, the jobs to be inserted into the existing schedule are grouped according to their due dates. Next, the heuristic determines the group with the least slack time and calculates penalties for each job in that group. Finally, the job with the minimum penalty is inserted into the sequence and the same steps are repeated until all the jobs are sequenced. Lofgren et al. [82] study the problem of routing PCBs through an assembly cell, where the sequence of the component assembly operations is determined so as to minimize the number of workstation visits. The authors use a graph theoretic approach in which a precedence graph is partitioned according to workstations and used to determine a workstation sequence which has the minimum cardinality. The problem is shown to be $\mathcal{NP}$–hard and the authors test and analyze different heuristics for the problem. Ben–Arieh and Maimon [20] consider a case with two sequential machines that use the same sequence (i.e., a permutation schedule). The objective is to minimize the mean flow time, and the authors solve the problem with a simulated annealing approach. Kim et al. [73] consider scheduling of PCBs in a flowshop manufacturing environment with the objective of minimizing the sum of tardiness. The authors compare four heuristics. The first is a modification of a previously proposed heuristic in the literature and sorts the jobs in nondecreasing order of due dates. The second heuristic is called the "extensive neighborhood search heuristic" and begins with

a proper initial sequence and tries to improve it with swap operations. The third heuristic is based on tabu search, and the last one on simulated annealing. Lin et al. [80] present a bi–level scheduling system (BLISS) which integrates both product level and board level scheduling on a capacitated multiple flowline and strives to determine economical lot sizes and board sequences. Product level scheduling shifts a complete set of boards for a product so that no work–in–process inventory is built up. Because this does not necessarily use the production capacity optimally, the schedule is then improved with a board level scheduling. The authors approach the problem by applying general scheduling methods from the literature. Based on experimental results they conclude that the two–level approach suits well when due dates are very tight or product demand is highly variable. Cunningham and Browne [31] argue that heuristic approaches are appropriate for the problem of sequencing PCBs by showing that the problem is $\mathcal{NP}$–hard. They propose heuristics where the PCBs are first divided into groups using similarity coefficients. Then, a branch–and–bound search technique is used to identify a good schedule for each group. Smed et al. [110] develop an interactive system for scheduling the PCB assembly processes. With the main objective of minimizing the setup times, they develop heuristics for PCB grouping, feeder optimization, and sequencing the component placement operations. The problem of identifying the sequence of board groups and sequences of board types within groups is left to the user. Häyrinen et al. [60] propose several scheduling algorithms for a generalized flexible flowshop environment in PCB assembly. The first heuristic addresses the problem of family allocation to machines, while the second heuristic tries to improve the allocation. A third heuristic is used to sequence the PCB families and the PCB types within the families and another heuristic is invoked to improve the sequence.

Of important notice is the fact that, when partial setup strategy is used, the setup time for the next board depends not just on its immediate predecessor but all of the preceding boards. This is due to the fact that the components loaded on a machine at a given instant depend on *all of the previous boards* and *the order they were processed.* In a sense, the setup time is *carried over* from the first board to the next board to be produced. This fact is recognized by Leon and Peters [75] (discussed previously above), but they use a myopic look–ahead approach to estimate the setup times rather than considering the actual setup times explicitly. Most recently, Rossetti and Stanford [102] recognize this issue as well. However, due to the complexity of the problem with this setup structure, they propose a heuristic approach based on estimating the expected number of setups that may occur given a board–feeder configuration, so that the problem is approximated by sequence–independent setup times. The estimates so obtained are used in nearest neighbor heuristics.

It should also be noted here that the setup strategy employed in this dissertation is a combination of the group setup strategy and the minimum setup strategy (also a combination of group setup strategy and the partial setup strategy since the partial setup strategy is a generalization of the minimum setup strategy). In minimum setup strategy the setups are performed in between *individual board types.* In group setup strategy discussed above, *all* the components are removed from the feeders and all the components required of the next *board group* are loaded on the appropriate feeders. However, the setup operations in our scheduling problems are performed in a way that only the components required of the next board group that are not present on the machine are loaded on the appropriate feeders (e.g., like it is performed in the minimum setup strategy for individual board types), while other components remain untouched. Therefore,

the setup time required of a board group depends not only on the immediately preceding board group but all of the preceding board groups and the order they were processed. In summary, the problems addressed in this dissertation are challenging problems that are extremely hard, but need to be solved efficiently and effectively. This dissertation not only develops highly efficient tabu search algorithms, but also quantifies the performance of the algorithms. Tight lower bounds on the optimal makespan and mean flow time are obtained via a column generation/branch–and–price algorithm and the quality of the solutions identified by the tabu search algorithms are quantified with respect to the lower bounds.

This chapter may have overlooked other research efforts reported in the published literature. For general reviews on the production planning and control problems encountered in PCB assembly, the reader is referred to Crama et al. [30], Ji and Wan [68], and McGinnis et al. [93]. For production planning and scheduling models in the semiconductor industry at large, refer to the two–part review by Uzsoy et al. [117, 118] and the references therein.

3. PROBLEM DESCRIPTION

This chapter describes the scheduling problem addressed in this dissertation and further illustrates it on a representative example taken from the industry. The problems with each of the two objectives (minimizing the makespan and minimizing the mean flow time) are shown to be $\mathcal{NP}$–hard in the strong sense.

The problem addressed here specifically relates to scheduling of the assembly processes required of PCBs in a multi–machine flowshop environment and is motivated by a real industry application encountered at a major electronics manufacturing company. As noted before, most of the time, two or three machines, arranged sequentially, are used to assemble PCBs. Figure 3.1 presents a two–machine PCB assembly system.

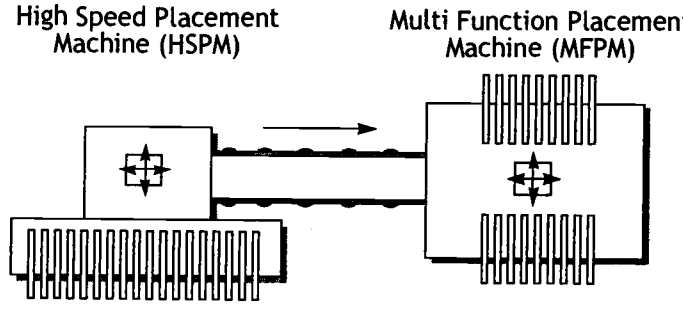

Figure 3.1. Two–Machine Assembly System

Machines 1 and 2 are referred to as the high–speed placement machine (HSPM) and multi–function placement machine (MFPM), respectively. Typically, these machines consist of 20 feeders (slots) on the magazine rack. The components required of the various boards to be produced are assigned to these feeders. On each machine, the pick–and–place device picks the components from the feeders and places them at prescribed locations on the boards.

Therefore, before inserting on the boards, the required components should be loaded on the appropriate feeders on the machines. It is this setup operation that requires a substantial amount of time compared to the time required for the machine to actually insert the components on the boards. The setup time on each machine is evaluated as the cumulative time it takes to perform four tasks:

1. The machine, when finishes processing a set of board types, should be brought to a complete stop.

2. The components, which are used in the assembly of the previous set of board types, should be removed and stacked up on the rack if the next set of board types require different components on the correponding feeders.

3. The components required of the next set of board types that are currently not loaded on the appropriate feeders must be taken off the rack and loaded on the appropriate feeders.

4. A complete check must be performed to make sure that all the required components are loaded on the appropriate feeders, and necessary programs must be loaded on the machine.

As discussed in the previous chapter, group technology (GT) principles are applied to the PCB assembly processes to reduce the setup times. PCBs with similar components are typically included in one group. Thus, the PCB assembly scheduling problem addressed in this dissertation falls under the category of *group-scheduling* and the issues concerning the PCB assembly must be addressed at two levels. At the first level or the individual board level, board types that belong to a group would have to be sequenced. At the second level or the board group level, the sequence of different board groups themselves would have to be determined.

In this research regarding the PCB assembly, unlike in flexible manufacturing systems (FMS), it is impractical to keep changing the component feeders on a frequent basis. Consistent with the GT principles, the resolve is to perform a setup that would enable placing all of the components required by several board types that belong to a single board group, which is referred to as the *surrogate* board group, so that a change from a board type to another in the same group requires no setup. However, as the groups themselves are dissimilar, the setup time required to switch production from one board group to another is so substantial that it cannot be disregarded. The setup on the second machine can be performed in anticipation of the arriving surrogate board group. This is a characteristic referred to as *anticipatory setups* in published research [4], meaning that the setup operation on a machine can start as soon as the machine becomes idle waiting for the next job that is still being processed on the previous machine (before the job actually arrives at the machine).

It is important to recognize that the characteristics of group–scheduling problems in electronics manufacturing are different from that of traditional manufacturing. Note that the setup strategy employed in this dissertation is a combination of the group setup strategy and the minimum setup strategy, which were discussed in the previous chapter. Hence, the time to perform the setup required of the current group of PCBs on a machine is not only sequence–dependent, but is also *carried over* from the first group to the current group. In other words, if N board groups are scheduled in the order $1, 2, \ldots, N$, the setup time needed to stage the components required of the N^{th} board group is dependent not just on the immediately preceding group (i.e., $(N-1)$) as it is in conventional sequence–dependent scheduling problems, but on *all* of the preceding groups (i.e., $1, 2, \ldots, (N-1)$) and the *order* they are sequenced. Thus, the problem addressed

in this dissertation is characterized as the "multi–machine carryover sequence–dependent group–scheduling problem with anticipatory setups." In addition, the challenges encountered in this research are far greater and distinctly different from those reported previously, including that on FMS. First, the setup change required of a board group is dependent upon the *component* and the *feeder location* to which it is allocated. Second, the assembly processes on PCBs are performed on multiple sequential machines.

3.1. A Representative Example

Tables 3.1–3.4 exhibit the data for the representative example problem obtained from the industry. Table 3.1 presents the boards to be produced on the two–machine assembly system depicted in Figure 3.1. Five different board types (G_{11}, G_{21}, G_{22}, G_{31}, and G_{32}) belong to three different board groups (G_1, G_2, and G_3). The number of each PCB type to be produced and their run times on both machines are also presented.

Table 3.1. An Example Problem with Three Board Groups

Groups	G_1	G_2		G_3	
Boards in a Group	G_{11}	G_{21}	G_{22}	G_{31}	G_{32}
Number of Boards to Build	5	7	3	14	9
Run Time on HSPM (sec.)	318	319	141	1096	1083
Run Time on MFPM (sec.)	20	41	3	488	355

Table 3.2 exhibits the average time required to load a single feeder on each machine, whereas Tables 3.3 and 3.4 present the feeder–component configuration

required of each board type on HSPM and MFPM, respectively. There are 20 feeders on HSPM and 10 feeders on MFPM, although a few are never used in this example. As noted before, the PCB scheduling process is typically broken down into several periods, each comprised of a shift or a day. The planning horizon will therefore consist of a series of periods. The initial/reference group, denoted by R, for the current period is given by the allocation of components to feeders at the end of the previous period.

Table 3.2. Average Feeder Setup Times on HSPM and MFPM

Machine	HSPM	MFPM
Average Setup Time per Feeder (sec.)	180	220

Observe that G_{11} forms its own (surrogate) group denoted by G_1, while G_{21} and G_{22} are two different board types that belong to the same group. Consistent with the similarity of the board types that belong to the same group, a majority of the components required of G_{21} and G_{22} are the same, while the remaining ones are different. Thus, the setup operation should not focus on either G_{21} or G_{22}, but rather on the surrogate representative of both G_{21} and G_{22}, which for simplicity is referred to as board group G_2. Clearly, this is a two–machine (HSPM and MFPM) group–scheduling problem, requiring sequencing decisions to be made at both levels. The challenge is to determine the sequence of board types in each group and the various board groups themselves in order to minimize the makespan/mean flow time.

Table 3.3. Feeder Configuration for the High–Speed Placement Machine

Reference Configuration on HSPM	Feeder	Feeder Configuration Required per Group			Feeder Configuration Required for Each Board Type				
		G_1	G_2	G_3	G_{11}	G_{21}	G_{22}	G_{31}	G_{32}
	1			101-02				101-02	101-02
101-06	2		101-04			101-04	101-04		
101-10	3	101-10	101-09		101-10	101-09			
101-01	4		101-01				101-01		
	5	101-08	101-05		101-08	101-05	101-05		
101-30	6	101-06	101-13	101-13	101-06		101-13	101-13	101-13
101-07	7	101-07	101-07	101-06	101-07	101-07		101-06	
101-13	8	101-13		101-03	101-13			101-03	101-03
101-33	9		101-08	101-08		101-08	101-08		101-08
	10	101-04			101-04				
101-35	11								
	12	101-09			101-09				
	13		101-06			101-06	101-06		
101-38	14	101-03	101-03	101-12	101-03	101-03	101-03		101-12
	15								
101-40	16	101-11	101-12		101-11		101-12		
	17								
101-11	18		101-11	101-09		101-11		101-09	101-09
	19		101-14	101-10		101-14		101-10	101-10
	20								

Table 3.4. Feeder Configuration for the Multi–Function Placement Machine

Reference Configuration on MFPM	Feeder	Feeder Configuration Required per Group			Feeder Configuration Required for Each Board Type				
		G_1	G_2	G_3	G_{11}	G_{21}	G_{22}	G_{31}	G_{32}
101-63	1	101-15		101-15	101-15			101-15	
101-84	2								
	3								
	4		101-15	101-18		101-15	101-15	101-18	101-18
	5	101-18	101-17		101-18	101-17			
101-18	6								
101-83	7			101-19				101-19	101-19
	8								
101-20	9		101-16				101-16		
	10			101-20				101-20	101-20

The computation of carryover sequence–dependent setup times on a machine can be demonstrated as follows. Consider the sequence $G_2(G_{21} - G_{22}) - G_1(G_{11}) - G_3(G_{31} - G_{32})$. As the setup involves surrogate groups, it is sufficient to consider the sequence of board groups (i.e., $G_2 - G_1 - G_3$) only. Clearly, three setups (one from R to G_2, one from $(R - G_2)$ to G_1, and one from $(R - G_2 - G_1)$ to G_3) are required. From Table 3.3, the number of setup changes on the feeders from R to G_2 on HSPM can be evaluated as follows. G_2 requires component 101-04 on feeder 2 but component 101-06 is loaded on feeder 2 in the reference configuration. Thus, a setup change is needed on feeder 2. G_2 demands component 101-01 on feeder 4. This component is already loaded on feeder 4. Thus, no setup is needed

on feeder 4. Following the same logic for all feeders on which G_2 requires a component, the total number of setup changes needed from R to G_2 is evaluated as 9 (1+1+0+1+1+0+1+1+1+1+0+1). The number of setup changes from $(R-G_2)$ to G_1 is evaluated in the same way with the difference that the feeders involving G_2 are loaded with the corresponding components while other feeders remain unchanged. For example, at feeder 3, G_1 requires component 101-10 but now it is component 101-09 loaded on feeder 3 as it was required by G_2. Therefore, one setup change is required on feeder 3. On the other hand, G_3 requires component 101-13 on feeder 8. Since component 101-13 was already loaded on feeder 8 at the beginning and G_2 did not require any change at feeder 8, no further setup change is required on feeder 8 for G_1. A total of 6 (1+1+1+0+0+1+1+0+1) setup changes are needed from $(R-G_2)$ to G_1, and 7 (1+1+1+1+0+1+1+1) from $(R-G_2-G_1)$ to G_3. The same mechanism also applies for MFPM. Recognizing that a setup on HSPM and MFPM takes 180 and 220 seconds, respectively, the setup time from R to G_2, for instance, is evaluated to be 1620 seconds on HSPM and 660 seconds on MFPM.

A pseudocode for computing the carryover sequence–dependent setup times for a given sequence on a given machine is also provided. In the algorithm presented in Figure 3.2 below, σ denotes a given sequence of board groups and board types within groups. In the development below and in the remainder of this dissertation, notations typically used in the scheduling literature are used. For example, if g denoted group g, (g) notation would denote the g^{th} group in the *rank ordered sequence*, i.e., g^{th} group in a given sequence σ. Similarly, if the g, b notation denoted the board type b belonging to board group g, the $(g), (b)$ notation would denote the b^{th} board type belonging to the g^{th} board group in the rank ordered sequence.

algorithm compute–setup–times

Inputs: σ: A sequence of board groups and board types within groups.
N: Number of board groups.
k: Index of the machine.
FS_k^g: Set of indices of feeders on which group g requires a component on machine k, $g = 0, \ldots, N$.
$CS_{g,k}^f$: The component required of group g in σ on machine k on feeder f, $g = 0, \ldots, N$; $f = 1, \ldots, FS_k^g$.
ST_k: Setup time per feeder on machine k.
Output: $S_{\sigma,k}[(g)]$: Setup time required of the g^{th} group in sequence σ on machine k, $g = 1, \ldots, N$.
Let $CurrentConfig[f]$ be the component currently loaded on feeder f, $f \in \cup_{g=0}^{N} FS_k^g$.

```
for all f ∈ FS_k^0 do
    CurrentConfig[f] ← CS_{g,k}^f          ▷ Current configuration on the machine
end for
for g ← 1 to N do
    S_{σ,k}[g] ← 0                         ▷ Initialize the setup time to 0
    for all f ∈ FS_k^g do
        if CurrentConfig[f] ≠ CS_{(g),k}^f then   ▷ If there is a different component
            S_{σ,k}[g] ← S_{σ,k}[g] + ST_k        ▷ Perform one feeder setup
            CurrentConfig[f] = CS_{(g),k}^f       ▷ Update current configuration
        end if
    end for
end for
```

Figure 3.2. Algorithm for Computing the Carryover Setup Times

3.2. Complexity of the Research Problem

This section shows that the problem under interest, with each of the two objectives, is $\mathcal{NP}$–hard in the strong sense. There are a number of techniques to prove that a problem is a member of a certain complexity class. The method used here is *proof by restriction* [44]. An $\mathcal{NP}$–hardness proof by restriction for a given

problem P consists of simply showing that P contains a known $\mathcal{NP}$–hard problem Q as a special case. The main point in this method lies in the specification of the additional restrictions to be placed on the instances of P so that the resulting restricted problem will be identical to Q.

First, it is essential to note that the carryover sequence–dependent setup times structure is a generalization of the (non–carryover) sequence–dependent setup times structure. When there is carryover sequence–dependency, the setup time required to switch production from group g to g' depends on the sequence of the groups. Therefore, *for every possible group sequence*, there is a certain amount of setup time required to switch from group g to g'. One can describe the non–carryover sequence–dependent setup time to switch from group g to group g' as carryover sequence–dependent by simply setting the same amount setup time to switch from group g to g' in all possible group sequences.

Theorem 3.2.1. *Multi–machine carryover sequence–dependent group–scheduling problem with the objective of minimizing the makespan is $\mathcal{NP}$–hard in the strong sense.*

Proof. Let P be the multi–machine (flowshop) carryover sequence–dependent group–scheduling problem with the objective of minimizing the makespan. Construct problem Q as a two–machine flowshop group–scheduling problem with the objective of minimizing the makespan and let each group consists of a single job. Also let the sequence–dependent setup times in problem Q be not defined to be carried over, but still be sequence–dependent. Clearly, problem Q is a special case of problem P (even when problem P involves more than two machines, since all the setup and run times on the machines other than the first two can be restricted to be 0 in problem P). Observe that problem Q is equivalent to the two–machine

flowshop scheduling problem with sequence–dependent setup times with the objective of minimizing the makespan, which has been shown to be $\mathcal{NP}$–hard in the strong sense [54]. Hence, problem P, which is a generalization of problem Q, is $\mathcal{NP}$–hard in the strong sense. □

Theorem 3.2.2. *Multi–machine carryover sequence–dependent group–scheduling problem with the objective of minimizing the mean flow time is $\mathcal{NP}$–hard in the strong sense.*

Proof. Let P be the multi–machine (flowshop) carryover sequence–dependent group–scheduling problem with the objective of minimizing the mean flow time. Construct problem Q as a two–machine flowshop group–scheduling problem with the objective of minimizing mean flow time where each group consists of a single job and let the setup times in problem Q be all 0. It is clear that problem Q is a special case of problem P even when the latter problem involves more than two machines, since all the setup and run times on the machines other than the first two can be restricted to be 0 in problem P. Observe that problem Q is equivalent to the two–machine flowshop scheduling problem with the objective of minimizing the mean flow time, which has been shown to be $\mathcal{NP}$–hard in the strong sense [45]. It follows that problem P is $\mathcal{NP}$–hard in the strong sense. □

4. MATHEMATICAL PROGRAMMING FORMULATIONS

This chapter proposes two novel modeling frameworks for formulating group–scheduling problems. These two frameworks can be used to formulate group–scheduling problems with different setup time structures such as sequence–independent or sequence–dependent setup times with or without carryover, or with no setups at all. Based on the frameworks, three different mixed–integer linear–programming (MILP) formulations of the carryover sequence–dependent setup times group–scheduling problem are presented. These formulations are compared in terms of the number of binary and continuous variables, and their ability to solve problems of various sizes.

In the literature, typically two modeling paradigms are used to formulate scheduling problems. The first was suggested by Wagner [119] and considers the order of a job within the sequence as a decision variable. The sequence of jobs is determined by assigning jobs to slots where each slot is indexed to represent the order of the job assigned to it. Second paradigm is due to Manne [92] and makes use of precedence relations between the jobs to determine their sequence. The first proposed formulation utilizes the first paradigm described above, while the second and third formulations incorporate the two paradigms in a linear fashion.

4.1. MILP1

The first MILP formulation utilizes the problem data as presented in section 3.1, directly using the component–feeder assignment tables to implicitly evaluate the carryover sequence–dependent setup times. The model determines a sequence of board types by assigning each board type to a slot while making sure that the board types within the same group are processed all together one after

the other. The setup times required of board groups in the optimal sequence is computed by the model by determining the values of the decision variables so that all of the required components are loaded on the appropriate feeders (by performing feeder setups) when necessary.

The following notation is adopted to present our mathematical formulations. Additional notation will be introduced when necessary in the later sections.

Indices and Parameters:

m = Number of machines.

k = $1, \ldots, m$ machines.

N = Number of board groups.

g, g' = $1, \ldots, N$ board groups.

n_g = Number of board types belonging to group g, $g = 1, \ldots, N$.

b = $1, \ldots, n_g$ board types, $g = 1, \ldots, N$.

J = Total number of board types, i.e., $J = \sum_{g=1}^{N} n_g$.

j = $1, \ldots, J$ slots.

$rt_{g,b,k}$ = Run time, in seconds, of board type b of board group g on machine k, $g = 1, \ldots, N$; $b = 1, \ldots, n_g$; $k = 1, \ldots, m$.

ST_k = Setup time, in seconds, per feeder on machine k, $k = 1, \ldots, m$.

FS_k = Set of indices of all required feeders on machine k, $k = 1, \ldots, m$.

$a_{g,k}^{c,f}$ = 1 if group g requires component c on feeder f on machine k, 0 otherwise, $g = 1, \ldots, N$; $k = 1, \ldots, m$; $f \in FS_k$; $c \in CS_k^f$.

CS_k^f = Set of indices of the components required on feeder f on machine k by all groups, $k = 1, \ldots, m$; $f \in FS_k$.

For clarity, FS_k and CS_k^f can be described using the data of the representative example problem presented on page 34 as follows. On machine 1, all the feeders are required by at least one board group, except for feeders 15, 17, and 20. Hence $FS_{k=1} = \{1,2,3,4,5,6,7,8,9,10,11,12,13,14,16,18,19\}$. Now, consider feeder 2 on machine 1. Components 101-04 and 101-06 are the only components required on feeder 2. Hence, $CS_{k=1}^{f=2} = \{101\text{-}04, 101\text{-}06\}$. Similarly $CS_1^5 = \{101\text{-}05, 101\text{-}08\}$ and $CS_1^6 = \{101\text{-}06, 101\text{-}13, 101\text{-}30\}$.

For machine 2, $FS_{k=2} = \{1,2,4,5,6,7,9,10\}$. Components 101-17 and 101-18 are the only components that can occupy feeder 2 during the production of all board types. Hence, $CS_{k=2}^{f=5} = \{101\text{-}17, 101\text{-}18\}$. Similarly, $CS_2^7 = \{101\text{-}19, 101\text{-}83\}$ and $CS_2^{10} = \{101\text{-}20\}$.

Decision Variables:

$x_{g,b,j}$ = 1 if board type b of group g is assigned to slot j, 0 otherwise,
$g = 1,\ldots,N;\ b = 1,\ldots,n_g;\ j = 1,\ldots,J.$

$y_{g,g',j}$ = 1 if a group setup is required from group g to g' at slot j, 0 otherwise,
$g = 1,\ldots,N;\ g' = 1,\ldots,N;\ g \neq g';\ j = 1,\ldots,J.$

$C_{j,k}$ = Completion time of the board type assigned to slot j on machine k,
$j = 1,\ldots,J;\ k = 1,\ldots,m.$

$h_{j,k}^{c,f}$ = 1 if component c is loaded on feeder f at slot j on machine k,
0 otherwise, $j = 1,\ldots,J;\ k = 1,\ldots,m;\ f \in FS_k;\ c \in CS_k^f$. Note that, the components of the initial configuration are loaded on the machines at the beginning, i.e., $h_{0,k}^{c,f} = a_{0,k}^{c,f}\ k = 1,\ldots,m; f \in FS_k, c \in CS_k^f$.

$d^f_{j,k}$ = 1 if a setup is needed on feeder f at slot j on machine k,

0 otherwise, $j = 1, \ldots, J$; $k = 1, \ldots, m$; $f \in FS_k$.

$S_{j,k}$ = Setup time required of the board type assigned to slot j on machine k,

$j = 1, \ldots, J$; $k = 1, \ldots, m$.

Following this notation, a MILP formulation for the problem with the objective of minimizing the *total flow time* can be presented as follows.

MILP1:

$$z^*_{MILP1} = \text{Min} \sum_{j=1}^{J} C_{j,m} \tag{4.1.1}$$

subject to

$$\sum_{g=1}^{N} \sum_{b=1}^{n_g} x_{g,b,j} = 1 \qquad \forall j \tag{4.1.2}$$

$$\sum_{j=1}^{J} x_{g,b,j} = 1 \qquad \forall g,\ b = 1, \ldots, n_g \tag{4.1.3}$$

$$\sum_{b=1}^{n_g} x_{g,b,j-1} + \sum_{b=1}^{n_{g'}} x_{g',b,j} \leq y_{g,g',j} + 1 \qquad \forall g,\ \forall g',\ g \neq g',\ j = 2, \ldots, J \tag{4.1.4}$$

$$\sum_{g=1}^{N} \sum_{g'=1, g \neq g'}^{N} \sum_{j=1}^{J} y_{g,g',j} = N - 1 \tag{4.1.5}$$

$$C_{j,k} \geq C_{j-1,k} + S_{j,k} + \sum_{g=1}^{N} \sum_{b=1}^{n_g} rt_{g,b,k} x_{g,b,j} \qquad \forall j,\ \forall k \tag{4.1.6}$$

$$C_{j,k} \geq C_{j,k-1} + \sum_{g=1}^{N} \sum_{b=1}^{n_g} rt_{g,b,k} x_{g,b,j} \qquad \forall j,\ k = 2, \ldots, m \tag{4.1.7}$$

$$h^{c,f}_{j,k} \geq \sum_{g=1}^{N} \sum_{b=1}^{n_g} a^{c,f}_{g,k} x_{g,b,j} \qquad \forall j, \forall k, f \in FS_k, c \in CS^f_k \tag{4.1.8}$$

$$h^{c,f}_{j,k} \leq h^{c,f}_{j-1,k} + \sum_{g=1}^{N} \sum_{b=1}^{n_g} a^{c,f}_{g,k} x_{g,b,j} \qquad \forall j, \forall k, f \in FS_k, c \in CS^f_k \tag{4.1.9}$$

$$\sum_{c \in CS_k^f} h_{j,k}^{c,f} \leq 1 \qquad \forall j, \forall k, f \in FS_k \qquad (4.1.10)$$

$$d_{j,k}^{f} \geq h_{j,k}^{c,f} - h_{j-1,k}^{c,f} \qquad \forall j, \forall k, f \in FS_k,\ c \in CS_k^f \qquad (4.1.11)$$

$$S_{j,k} = \sum_{f \in FS_k} ST_k\, d_{j,k}^{f} \qquad \forall j,\ \forall k \qquad (4.1.12)$$

$$x_{g,b,j} \in \{0,1\} \qquad \forall g,\ b = 1, \ldots, n_g,\ \forall j \qquad (4.1.13)$$

$$y_{g,g',j} \in \{0,1\} \qquad \forall g, \forall g',\ g \neq g',\ \forall j \qquad (4.1.14)$$

$$h_{j,k}^{c,f} \in \{0,1\} \qquad \forall j,\ \forall k,\ f \in FS_k,\ c \in CS_k^f \qquad (4.1.15)$$

$$d_{j,k}^{f} \in \{0,1\} \qquad \forall j,\ \forall k,\ f \in FS_k \qquad (4.1.16)$$

The objective (4.1.1) is to minimize the total flow time, which is the sum of the completion times of the individual board types (jobs) on the last machine. It can be divided by the total number of board types, J, in order to minimize the *mean* flow time. For minimizing the *makespan*, the objective should be changed to minimizing the completion time of the last board type, which is equal to the completion time of the board type that is assigned to the last slot on the last machine, i.e., $C_{J,m}$.

Constraint sets (4.1.2) and (4.1.3) are the assignment constraints. They make sure that each board type is assigned to exactly one slot, and each slot is assigned to exactly one board type. Constraint set (4.1.4) determines which setup operation should be performed on the slots other than the first slot. By ensuring that the production switches from a group to another $N - 1$ times (excluding the setup for the first group), constraint set (4.1.5) ensures that the board types (jobs) within a group are processed all together, one after the other. Constraint set (4.1.6) evaluates the completion times of the board types assigned to each slot *on each machine alone.* On each machine, the completion time of the board type

assigned to a slot, say slot j, is at least the completion time of the board type assigned to the preceding slot (slot $j-1$) plus the setup time in between plus the run time of the board type assigned to slot j. Constraint set (4.1.7) describes the relationship between the machines, i.e., it ensures that the completion time of a board type on the current machine is at least its completion time on the previous machine plus its run time on the current machine. Therefore, this constraint set ensures that a board type starts operation on a machine only after it is completed on the previous machine. Constraint set (4.1.8) makes sure that all the components required of the board group that is assigned to the current slot are loaded on the appropriate feeders. Constraint set (4.1.9), on the other hand, makes sure that the component loaded on a feeder at the previous slot remains on that feeder at the current slot if the current board group does not require a component on that feeder. Constraint set (4.1.10) makes sure that, on any machine and at a given slot, a feeder is loaded with at most one component. Note that if a feeder is already loaded with a component at the initial/reference configuration, it will stay loaded (either with the same component of the reference group or with another component required of the board groups to be produced). If not, the constraint set ensures that there will be at most one component at that feeder during the production of all of the board groups. Constraint set (4.1.11) determines, for each feeder, if a setup is performed in between slots by checking whether a component is loaded on a feeder and was not loaded on that feeder at the previous slot. For each machine, constraint set (4.1.12) evaluates the setup time required before each slot by first evaluating the total number of feeder setups and then by multiplying that number by the average setup time per feeder on the machine.

Finally, (4.1.13)−(4.1.16) are the integrality restrictions on the decision variables.

4.2. MILP2

Remember that the first modeling framework proposed here assigns each of the individual board types to one of the J slots but makes sure that the board types within a group are processed contiguously, one after the other. The second framework, on the other hand, assigns each group to one of N slots, while the sequence of individual board types within each group is determined by constructing precedence constraints among the completion times of board types. Therefore, the novelty of this modeling framework is that it blends the two modeling paradigms proposed previously in the literature for formulating scheduling problems. The challenge is to establish a relationship between the order of a group and the order of the board types within that group while incorporating the two paradigms, and still preserve linearity.

Since which group will be assigned to which slot in the optimal sequence is not known a priori, it is not straightforward to assess the number of board types of the group assigned to a slot as the number of board types within board groups may be different. A remedy to this difficulty is to introduce 'dummy' board types just so that every group is composed of the same number of board types. For example, suppose that there are three board groups G_1, G_2, and G_3 with 1, 2, and 4 board types, respectively. Then, three more board types are added to G_1 and two more board types to G_2. Therefore, all the board groups now have 4 board types. However, the completion times of the dummy board types should be excluded from the evaluation of the makespan/mean flow time.

To present the second formulation, a notation slightly different than the one introduced in the first formulation is needed. Although some of the notation is the same as before, they are repeated here for clarity.

Indices and Parameters:

H = A very large positive number.

m = Number of machines.

k = $1, \ldots, m$ machines.

N = Number of board groups.

g = $1, \ldots, N$ board groups.

j = $1, \ldots, N$ slots.

n_g = Number of board types belonging to group g, $g = 1, \ldots, N$.

$nmax$ = Maximum number of board types in board groups,

i.e., $nmax = \max_{g=1,\ldots,N} \{n_g\}$.

b = $1, \ldots, nmax$ board types.

$rt_{g,b,k}$ = Run time of board type b of board group g on machine k, $\forall g, \forall b, \forall k$.

By definition, the run times of dummy board types are set to $-H$,

i.e., $rt_{g,b,k} = -H$ for $b = n_g+1, \ldots, nmax$, $\forall g$, $\forall k$.

ST_k = Setup time, in seconds, per feeder on machine k, $k = 1, \ldots, m$.

FS_k = Set of indices of all required feeders on machine k, $k = 1, \ldots, m$.

$a_{g,k}^{c,f}$ = 1 if group g requires component c on feeder f on machine k,

0 otherwise, $g = 1, \ldots, N$; $k = 1, \ldots, m$; $f \in FS_k$, $c \in CS_k^f$.

CS_k^f = Set of indices of the components required on feeder f on machine k

by all groups, $k = 1, \ldots, m$; $f \in FS_k$.

Decision Variables:

$x_{g,j}$ = 1 if board group g is assigned to slot j, 0 otherwise, $\forall g$, $\forall j$.

$y_{j,b,b'}$ = 1 if board type b' follows board type b in the group assigned to slot j of the group assigned to the same slot, 0 otherwise, $\forall b$, $\forall b'$, $b' > b$, $\forall j$.

$C_{j,k}$ = Completion time of the board group assigned to slot j on machine k, 0 otherwise, $\forall j$, $\forall k$.

$O_{j,b,k}$ = Completion time of board type b of the group assigned to slot j on machine k, $\forall j$, $\forall b$, $\forall k$.

$S_{j,k}$ = Setup time required of the board group that is assigned to slot j on machine k, $\forall j$, $\forall k$.

$h_{j,k}^{c,f}$ = 1 if component c is loaded on feeder f at slot j on machine k, 0 otherwise, $j = 1, \ldots, J$; $k = 1, \ldots, m$; $f \in FS_k$, $c \in CS_k^f$. Note that, only the components of the reference configuration are loaded on the machines at the beginning, i.e., $h_{0,k}^{c,f} = a_{0,k}^{c,f}$ $\forall k$, $f \in FS_k$, $c \in CS_k^f$.

$d_{j,k}^f$ = 1 if a setup is needed on feeder f at slot j on machine k, 0 otherwise, $\forall j$, $\forall k$, $f \in FS_k$.

Following this notation, a MILP formulation for the problem with the objective of minimizing the *total flow time* can be presented as follows.

MILP2:

$$z^*_{MILP2} = \text{Min} \sum_{j=1}^{N} \sum_{b=1}^{nmax} O_{j,b,m} \tag{4.2.1}$$

subject to

$$\sum_{g=1}^{N} x_{g,j} = 1 \qquad \forall j \tag{4.2.2}$$

$$\sum_{j=1}^{N} x_{g,j} = 1 \qquad \forall g \tag{4.2.3}$$

$$C_{j,k} \geq C_{j-1,k} + S_{j,k} + \sum_{g=1}^{N} \sum_{b=1}^{nmax} rt_{g,b,k} x_{g,j} \qquad \forall j,\ \forall k,\ C_{0,k} = 0 \qquad (4.2.4)$$

$$O_{j,b,k} \geq C_{j-1,k} + S_{j,k} + \sum_{g=1}^{N} rt_{g,b,k} x_{g,j} \qquad \forall j,\ \forall k,\ \forall b,\ C_{0,k} = 0 \qquad (4.2.5)$$

$$O_{j,b,k} \geq O_{j,b,k-1} + \sum_{g=1}^{N} rt_{g,b,k} x_{g,j} \qquad \forall j,\ \forall b,\ k = 2, \ldots, m \qquad (4.2.6)$$

$$C_{j,k} = \max_{b=1,\ldots,nmax} O_{j,b,k} \qquad \forall j,\ \forall k \qquad (4.2.7)$$

$$O_{j,b,k} - O_{j,b',k} + H y_{j,b,b'} \geq \sum_{g=1}^{N} rt_{g,b,k} x_{g,j} \qquad \forall j,\ \forall b,\ \forall b',\ b' > b, \forall k \qquad (4.2.8)$$

$$O_{j,b',k} - O_{j,b,k} + H(1 - y_{j,b,b'}) \geq \sum_{g=1}^{N} rt_{g,b',k} x_{g,j} \qquad \forall j,\ \forall b,\ \forall b',\ b' > b, \forall k \qquad (4.2.9)$$

$$h_{j,k}^{c,f} \geq \sum_{g=1}^{N} a_{g,k}^{c,f} x_{g,j} \qquad \forall j, \forall k, f \in FS_k, c \in CS_k^f \qquad (4.2.10)$$

$$h_{j,k}^{c,f} \leq h_{j-1,k}^{c,f} + \sum_{g=1}^{N} a_{g,k}^{c,f} x_{g,j} \qquad \forall j, \forall k, f \in FS_k, c \in CS_k^f \qquad (4.2.11)$$

$$\sum_{c \in CS_k^f} h_{j,k}^{c,f} \leq 1 \qquad \forall j, \forall k, f \in FS_k \qquad (4.2.12)$$

$$d_{j,k}^{f} \geq h_{j,k}^{c,f} - h_{j-1,k}^{c,f} \qquad \forall j, \forall k, f \in FS_k, c \in CS_k^f \qquad (4.2.13)$$

$$S_{j,k} = \sum_{f \in FS_k} ST_k\, d_{j,k}^{f} \qquad \forall j,\ \forall k \qquad (4.2.14)$$

$$x_{g,j} \in \{0, 1\} \qquad \forall g,\ \forall j \qquad (4.2.15)$$

$$y_{j,b,b'} \in \{0, 1\} \qquad \forall j, \forall b, \forall b',\ b' > b \qquad (4.2.16)$$

$$h_{j,k}^{c,f} \in \{0, 1\} \qquad \forall j, \forall k, f \in FS_k, c \in CS_k^f \qquad (4.2.17)$$

$$d_{j,k}^{f} \in \{0, 1\} \qquad \forall j,\ \forall k,\ f \in FS_k \qquad (4.2.18)$$

The objective (4.2.1) is to minimize the sum of the flow times, which can be divided by the total number of board types to minimize the *mean* flow time.

For minimizing the *makespan*, the objective function should be changed to the completion time of the last board group on the last machine, i.e., $C_{N,m}$.

Constraint sets (4.2.2) and (4.2.3) ensure that each board group is assigned to exactly one slot, and each slot is assigned to exactly one group. Constraint (4.2.4) schedules the board groups on each machine, making sure that the completion time of the current board group is at least the completion time of the preceding group on the same machine plus the setup time required of the current board group plus the entire run time of the current board group. Constraint set (4.2.5) makes sure that the completion time of an individual board type is at least the completion time of the preceding board group on the same machine plus the setup time required of the current board group plus the run time of the individual board type. Constraint set (4.2.6) ensures that a board type starts processing on a machine only after it is completed on the previous machine. Constraint set (4.2.7) sets the completion time of a board group to the completion time of the board type that is processed last in that group. Constraint sets (4.2.8) and (4.2.9) schedule the individual board types within each group, ensuring that the difference between the completion times of two board types is at least the run time of the board type scheduled to be processed later. Constraint set (4.2.10) makes sure that all the components required of the board group that is assigned to a slot are loaded on the appropriate feeders. Constraint set (4.2.11), on the other hand, ensures that the component loaded on a feeder at the previous slot remains on that feeder if the next board group does not require a component on that feeder. Constraint set (4.2.12) makes sure that, on any machine and at a given slot, a feeder is loaded with at most one component. Note that if a feeder is already loaded with a component at the initial/reference configuration, it will stay loaded (either with the same component of the reference group or with another

component required of the board groups to be produced). If not, the constraint set ensures that there will be at most one component at that feeder during the production of all of the board groups. Constraint set (4.2.13) determines if a setup is performed in between slots at each feeder by checking whether a component is loaded on a feeder and was not loaded on that feeder at the previous slot. Constraint set (4.2.14) evaluates the setup time, for each machine, required before each slot by first evaluating the total number of feeder setups for each slot and then by multiplying that number by the average setup time per feeder required on the machine.

Finally, (4.2.15)–(4.2.18) are the integrality restrictions on the decision variables.

Observe that, because the run times for the dummy board types are defined to be $-H$ and the sense of the objective is minimization, the completion time variables $O_{j,b,k}$ will be evaluated as 0 for dummy board types (observe the constraint sets (4.2.5), (4.2.6), (4.2.8), and (4.2.9)). Consequently, the dummy board types will have no contribution to the objective function value.

4.3. MILP3

Using the feeder carrier configuration required per group (see Tables 3.3 and 3.4 in section 3.1), constraint sets (4.1.8)–(4.1.12) in MILP1 and sets (4.2.10)–(4.2.14) in MILP2 implicitly compute the carryover sequence–dependent setup time required before starting production of each board group. However, given a sequence of board groups, the setup times can be computed explicitly as shown in Figure 3.2 previously. The third formulation proposed here assumes that the carryover setup times for each possible board group sequence have been computed

beforehand. As in MILP2, the optimal board group sequence is determined by assigning each group to one of N slots, and the sequence of individual board types within each group is determined by constructing precedence constraints among the completion times of board types.

Besides some of the indices and parameters defined for MILP2 previously, the following special constructs are introduced to aid in the development of the model.

$\Pi(j)$ = Set of all possible partial group sequences including exactly j groups excluding the reference group, R, which is denoted by group 0. Each partial group sequence begins with the reference group. For example, $\Pi(1) = \{(0-1), (0-2), \ldots, (0-N)\}$. $\Pi(0) = 0$ by definition.

π_j = A partial group sequence with exactly j groups, excluding group 0. For instance, $\pi_2 = (0-2-3)$. $\pi_0 = (0)$ by definition.

$\pi_j \oplus g$ = The partial group sequence constructed by attaching group g to the end of partial sequence π_j. E.g., $\pi_1 = (0-3) \Rightarrow \pi_1 \oplus 1 = (0-3-1)$.

$\pi_j \setminus k$ = The partial sequence composed of first k groups of the partial group sequence π_j, in the same order. For example, $\pi_2 = (0-2-3) \Rightarrow$ $\pi_2 \setminus 1 = (0-2)$ and $\pi_2 \setminus 2 = (0-2-3)$.

$\pi_j(k)$ = k^{th} group of the partial sequence π_j. For example, $\pi_2 = (0-2-3) \Rightarrow$ $\pi_2(1) = 2$, $\pi_2(2) = 3$.

$\psi(\pi_j)$ = The set of groups in the partial sequence π_j. For instance, $\pi_1 = (0-3) \Rightarrow \psi(\pi_1) = \{3\}$; $\pi_3 = (0-2-1-3) \Rightarrow \psi(\pi_3) = \{1,2,3\}$.

$S_{\pi_j,g,k}$ = Setup time required to process board group g right after the partial sequence π_j on machine k, $\pi_j \in \Pi(j)$, $j = 1, \ldots, N-1$; $\forall g$, $\forall k$.

Decision Variables:

$x_{g,j}$ = 1 if board group g is assigned to slot j, 0 otherwise, $\forall g$, $\forall j$.

$y_{j,b,b'}$ = 1 if board type b' of the group assigned to slot j follows board type b of the group assigned to the same slot, 0 otherwise, $\forall b$, $\forall b'$, $b' \geq b$, $\forall j$.

$C_{j,k}$ = Completion time of the group assigned to slot j on machine k, $\forall j$, $\forall k$.

$O_{j,b,k}$ = Completion time of board type b of the group assigned to slot j on machine k, $\forall j$, $\forall b$, $\forall k$.

$S_{j,k}$ = Carryover sequence–dependent setup time required of the board group assigned to slot j on machine k, $\forall j$, $\forall k$.

z_{π_N} = 1 if group sequence is identified to be π_N, 0 otherwise, $\pi_N \in \Pi(N)$.

Following this notation, a MILP formulation for the problem with the objective of minimizing the *total flow time* can be presented as follows.

MILP3:

$$z^*_{MILP3} = \text{Min} \sum_{j=1}^{N} \sum_{b=1}^{nmax} O_{j,b,m} \tag{4.3.1}$$

subject to

$$\sum_{\pi_N \in \Pi(N)} z_{\pi_N} = 1 \tag{4.3.2}$$

$$x_{g,j} = \sum_{\substack{\pi_N \in \Pi(N) \\ \pi_N(j)=g}} z_{\pi_N} \qquad \forall g, \forall j \tag{4.3.3}$$

$$C_{j,k} \geq C_{j-1,k} + S_{j,k} + \sum_{g=1}^{N} \sum_{b=1}^{nmax} rt_{g,b,k} x_{g,j} \qquad \forall j, \forall k, C_{0,k} = 0 \tag{4.3.4}$$

$$O_{j,b,k} \geq C_{j-1,k} + S_{j,k} + \sum_{g=1}^{N} rt_{g,b,k} x_{g,j} \qquad \forall j, \forall k, \forall b, C_{0,k} = 0 \tag{4.3.5}$$

$$O_{j,b,k} \geq O_{j,b,k-1} + \sum_{g=1}^{N} rt_{g,b,k} x_{g,j} \qquad \forall j, \forall b,\ k = 2, \ldots, m \tag{4.3.6}$$

$$C_{j,k} = \max_{b=1,\ldots,nmax} O_{j,b,k} \qquad \forall j, \forall k \qquad (4.3.7)$$

$$O_{j,b,k} - O_{j,b',k} + Hy_{j,b,b'} \geq \sum_{g=1}^{N} rt_{g,b,k} x_{g,j} \qquad \forall j, \forall b, \forall b',\ b' > b, \forall k \qquad (4.3.8)$$

$$O_{j,b',k} - O_{j,b,k} + H(1 - y_{j,b,b'}) \geq \sum_{g=1}^{N} rt_{g,b',k} x_{g,j} \qquad \forall j, \forall b, \forall b',\ b' > b, \forall k \qquad (4.3.9)$$

$$S_{j,k} = \sum_{\pi_{j-1} \in \Pi(j-1)} \sum_{\substack{g=1 \\ g \notin \psi(\pi_{j-1})}} S_{\pi_{j-1},g,k} \Big(\sum_{\substack{\pi_N \in \Pi(N) \\ \pi_N \setminus j = \pi_{j-1} \oplus g}} z_{\pi_N} \Big) \qquad \forall j, \forall k \qquad (4.3.10)$$

$$x_{g,j} \in \{0,1\} \qquad \forall g, \forall j \qquad (4.3.11)$$

$$y_{j,b,b'} \in \{0,1\} \qquad \forall j, \forall b, \forall b',\ b' > b \qquad (4.3.12)$$

$$z_{\pi_N} \in \{0,1\} \qquad \pi_N \in \Pi(N) \qquad (4.3.13)$$

The objective (4.3.1) is to minimize the sum of the flow times, which can be divided by the total number of board types to minimize the *mean* flow time. For minimizing the *makespan*, the objective function should be changed to the completion time of the last board group on the last machine, i.e., $C_{N,m}$.

Constraint (4.3.2) ensures that only one complete group sequence is identified. Constraint set (4.3.3) enforces that each board group is assigned to exactly one slot and this assignment is consistent with the group sequence identified. Constraint set (4.3.4) schedules the board groups on each machine. It ensures that the completion time of the current board group is at least the completion time of the preceding group on the same machine plus the setup time required of the current board group plus the entire run time of the current board group. Constraint set (4.3.5) makes sure that the completion time of an individual board type is at least the completion time of the preceding board group on the same machine plus the setup time required of the current board group plus the run time of the individual board type. Constraint set (4.3.6) ensures that a board type starts processing

on a machine only after it is completed on the previous machine. Constraint set (4.3.7) sets the completion time of a board group to the completion time of the board type that is processed last in that group. Constraint sets (4.3.8) and (4.3.9) schedule the individual board types within each group, ensuring that the difference between the completion times of two board types is at least the run time of the board type scheduled to be processed later. Constraint set (4.3.10) evaluates the appropriate carryover sequence–dependent setup times, depending on the complete group sequence identified. It simply states that the setup time required before each slot on each machine should be the ones corresponding to the optimal group sequence.

Finally, (4.3.11)–(4.3.13) are the integrality restrictions on the decision variables.

For clarity, the construction of carryover sequence–dependent setup times constraints is demonstrated on the representative example problem. The data involving carryover setup times for the two–machine example presented in Section 3.1 are given in Table 4.1. For all possible partial sequences at each slot, carryover sequence–dependent setup times are computed as described previously. Table 4.1 simply states, for example, that if group 3 were to be processed first and group 2 second, it would require 1620 seconds to setup machine 1 (HSPM) to switch production from group 3 to group 2 ($S_{(0-3),2,1} = 1620$). Similarly, if groups 2 and 3 were processed in that order and group 1 were to be processed next, the setup time would be 220 seconds on machine 2 ($S_{(0-2-3),1,2} = 220$).

Construction of the constraints for the carryover sequence–dependent setup times can be explained as follows. Define six z variables, one for each of six ($N! = 3! = 6$) possible group sequences, designated as $z_{(0-1-2-3)}$, $z_{(0-1-3-2)}$, $z_{(0-2-1-3)}$, $z_{(0-2-3-1)}$, $z_{(0-3-1-2)}$, and $z_{(0-3-2-1)}$. The setup times only on machine 1 will be

Table 4.1. Carryover Sequence–Dependent Setup Times of the Example Problem

		Machine 1			Machine 2		
Slot	Partial	Group g			Group g		
j	Sequence π_{j-1}	1	2	3	1	2	3
1	(0)	1080	1620	1440	440	660	880
2	(0–1)	-	1440	1440	-	660	660
	(0–2)	1080	-	1080	440	-	880
	(0–3)	1440	1620	-	660	660	-
3	(0–1–2)	-	-	1080	-	-	660
	(0–1–3)	-	1620	-	-	660	-
	(0–2–1)	-	-	1260	-	-	660
	(0–2–3)	1620	-	-	220	-	-
	(0–3–1)	-	1440	-	-	660	-
	(0–3–2)	1260	-	-	220	-	-

considered, and the group sequence $(0-3-1-2)$ will be used as an example to imply that $z_{(0-3-1-2)} = 1$, $x_{3,1} = 1$, $x_{1,2} = 1$, and $x_{2,3} = 1$. Since one of the six possible sequences should be identified, constraint set (4.3.2) takes the form:

$$z_{(0-1-2-3)} + z_{(0-1-3-2)} + z_{(0-2-1-3)} + z_{(0-2-3-1)} + z_{(0-3-1-2)} + z_{(0-3-2-1)} = 1.$$

For slot 1:

Initial partial group sequence is '(0)' (i.e., the reference group only). For slot 1, the only decision is to choose the next group. In order to identify the group

assigned to slot 1, constraint set (4.3.3) takes the following form:

$$x_{1,1} = z_{(0-1-2-3)} + z_{(0-1-3-2)},$$

$$x_{2,1} = z_{(0-2-1-3)} + z_{(0-2-3-1)},$$

$$x_{3,1} = z_{(0-3-1-2)} + z_{(0-3-2-1)}.$$

The setup time for the first slot/group is computed by the constraint (constraint set (4.3.10) for $j = 1$)

$$S_{1,1} = S_{(0),1,1}(z_{(0-1-2-3)} + z_{(0-1-3-2)}) + S_{(0),2,1}(z_{(0-2-1-3)} + z_{(0-2-3-1)}) + S_{(0),3,1}(z_{(0-3-1-2)} + z_{(0-3-2-1)}).$$

For the example sequence, $S_{1,1} = S_{(0),3,1} = 1440$ since only $z_{(0-3-1-2)} = 1$.

For slot 2:

The same set of constraints, except that the partial sequence is now composed of the reference group followed by the group assigned to slot 1, is introduced:

$$x_{1,2} = z_{(0-2-1-3)} + z_{(0-3-1-2)},$$

$$x_{2,2} = z_{(0-1-2-3)} + z_{(0-3-2-1)},$$

$$x_{3,2} = z_{(0-1-3-2)} + z_{(0-2-3-1)}.$$

The appropriate setup time for the second slot/group is computed by the following constraint (constraint set (4.3.10) for $j = 2$)

$$S_{2,1} = S_{(0-1),2,1}z_{(0-1-2-3)} + S_{(0-1),3,1}z_{(0-1-3-2)} + S_{(0-2),1,1}z_{(0-2-1-3)} + S_{(0-2),3,1}z_{(0-2-3-1)} + S_{(0-3),1,1}z_{(0-3-1-2)} + S_{(0-3),2,1}z_{(0-3-2-1)}.$$

For the example sequence $S_{2,1} = S_{(0-3),1,1} = 1440$ since only $z_{(0-3-1-2)} = 1$.

For slot 3:

Similarly, for the third slot/group, we have

$$x_{1,3} = z_{(0-2-3-1)} + z_{(0-3-2-1)},$$

$$x_{2,3} = z_{(0-1-3-2)} + z_{(0-3-1-2)},$$

$$x_{3,3} = z_{(0-1-2-3)} + z_{(0-2-1-3)}.$$

The appropriate setup time for the second slot/group is computed by the following constraint (constraint set (4.3.10) for $j = 3$)

$$S_{3,1} = S_{(0-1-2),3,1} z_{(0-1-2-3)} + S_{(0-1-3),2,1} z_{(0-1-3-2)} + S_{(0-2-1),3,1} z_{(0-2-1-3)}$$
$$+ S_{(0-2-3),1,1} z_{(0-2-3-1)} + S_{(0-3-1),2,1} z_{(0-3-1-2)} + S_{(0-3-2),1,1} z_{(0-3-2-1)}.$$

For the example sequence, $S_{3,1} = S_{(0-3-1),2,1} = 1440$ since only $z_{(0-3-1-2)} = 1$.

4.4. Comparison of the Proposed Formulations

Table 4.2 present the number of binary and continuous variables that exist in MILP1, MILP2 and MILP3.

Table 4.2. Number of Variables in MILP1, MILP2, and MILP3

MILP1	Binary	$J^2 + N(N-1)(J-1) + J\sum_{k=1}^{m}\left(\lvert FS_k\rvert + \sum_{f\in FS_k}\lvert CS_k^f\rvert\right)$
	Continuous	$2Jm$
MILP2	Binary	$N^2 + \sum_{g=1}^{N} n_g(n_g-1)/2 + J\sum_{k=1}^{m}\left(\lvert FS_k\rvert + \sum_{f\in FS_k}\lvert CS_k^f\rvert\right)$
	Continuous	$2Nm$
MILP3	Binary	$N! + N^2 + J(nmax-1)nmax/2$
	Continuous	$2Nm(nmax+1)$

Since a binary variable is defined for each possible group sequence, the total number of binary variables in MILP3 grows exponentially with the number of groups. Hence, it is impractical to use MILP3 for large size problems, which would require excessive amounts of computation memory. Problems with more than 8 board groups could not be solved with MILP3 due to lack of enough computation memory. However, for small and medium size problem instances (with 3 to 8 board groups), it has been experienced that MILP3 can be solved much faster than MILP1 or MILP2 with a branch–and–bound algorithm. It is conjectured that branching on z_{π_N} variables is very effective. The z_{π_N} variables in the constraint set (4.3.2) form a *type 3 special ordered set* (SOS Type 3) [39]. In a SOS Type 3 set, only one of the variables is allowed to be 1, while all the other variables have to be 0. It is well–known that *SOS branching*, i.e., branching on several of the variables in a SOS set at the same time rather than branching on the variables one by one, is generally more effective since the branch–and–bound tree is more *balanced*. Furthermore, in a node deep down the tree, setting one of the z_{π_N} variables to 1 implies that all the remaining z_{π_N} variables are fixed to 0 in that node. Hence, the problem for that node reduces to scheduling the individual board types within each group (since the group sequence is fixed), which is relatively easier.

The constraint sets (4.2.8) and (4.2.9) in MILP2 and (4.3.8) and (4.3.9) in MILP3, designed to schedule the individual board types within each group, are not binding for an optimal solution. In general, this loosens the linear–programming (LP) relaxation of the formulations. However, the fact that a binary variable is defined for each possible group sequence strengthens the LP–relaxation of MILP3. Hence, the LP–relaxation of MILP3 is in general tighter than that of MILP2. In addition, MILP2 and MILP3 require the introduction of 'dummy' board types

with run times set to $-H$ on all machines. This distorts the structure of the subproblems in a column generation algorithm that is designed based on MILP2 or MILP3. Although the run times of the dummy board types can be defined in a different way (see, for instance, Gelogullari and Logendran [43]), a column generation algorithm designed for MILP1 is more effective compared to that of MILP2 or MILP3.

In summary, MILP3 seems to be a good choice to solve small size problems with a commercial MILP solver, but cannot be used for large size problems. The LP–relaxation of MILP2 is too inferior compared to that of MILP1 because of the constraint sets (4.2.8) and (4.2.9). Hence, the column generation/branch–and–price algorithms are based on MILP1, which is more effective and can handle large size problems.

The computational experiments presented in chapter 8 reveal the differences between the LP–relaxations of these formulations and their ability to identify optimal solutions or integer feasible solutions for problems of various sizes.

5. TABU SEARCH ALGORITHMS

Since the research problems, with each of the two objectives, are $\mathcal{NP}$–hard in the strong sense, it is highly likely that existing optimization algorithms that guarantee global optima will require excessive amounts of computation time. Hence, high level metasearch heuristic algorithms are proposed to identify good quality solutions. The proposed search algorithms are not only time–efficient but also effective with respect to identifying good quality solutions even on large, industry–size problems encountered in electronics manufacturing.

This chapter presents the structure of *novel* search algorithms based on the concept known as *tabu search*. First, the necessary background is developed and then the proposed tabu search algorithms are described in detail.

5.1. Introduction

Tabu Search (TS) is a fairly new method that is still actively researched and is continuing to evolve and improve. The roots of TS go back to late 1970's (Glover [47]). It was first presented in its current form by Glover [48]; the basic ideas have also been sketched by Hansen [57]. Additional reports to formalize the TS approach include Glover [49–51], Glover and Laguna [52], and De Werra and Hertz [33]. Recently, Glover and Laguna [53] edited a comprehensive book that presents historical origins of TS and different features it adapted from a variety of fields.

TS has been proven to be a remarkably successful approach for solving a wide range of hard combinatorial optimization problems. Barnes et al. [17] review applications of TS to various production scheduling problems. It is successfully applied to complex machine scheduling problems including that of minimization of

the makespan of a flexible manufacturing system (Logendran and Sonthinen [86]) and of a real–time scheduling problem on unrelated parallel machines with job splitting (Logendran and Subur [88]).

TS is a general framework specifically designed to guide any other method such as local search (hence the name "metasearch heuristic" or, in short, "meta-heuristic"). It is an iterative method that starts from an initial solution and marches towards better solutions, employing a set of moves for transforming one solution to another. TS is founded on the following three primary themes.

- The use of *adaptive memory* structures designed to permit historical search information to explore the solution space more economically and effectively than by rigid memory structures as in branch–and–bound methods or memoryless systems such as simulated annealing or genetic algorithms.
- A control mechanism of *responsive exploration* of the solution space that constrains the search by classifying certain of its moves as forbidden (i.e., "tabu") and that frees the search to provide "strategic forgetting."
- The incorporation of memory functions of different time spans to *intensify* the search within the regions of the solution space containing promising solutions or *diversify* the search towards less explored regions of the solution space.

The emphasis of responsive exploration in TS derives from the supposition that a bad strategic choice can yield more information than a good random choice. A bad choice can provide useful clues about how the search strategy may profitably be changed; a purposeful design can be more adept at uncovering the imprint of structure in elite solutions. The memory used in TS is both *explicit*

and *attributive*. Explicit memory records complete solutions, typically consisting of elite solutions visited during the search. The search can be expanded around the memorized elite solutions. Alternatively, TS uses attributive memory for guiding purposes. This type of memory records information about solution attributes that change in moving from one solution to another. For example, in production scheduling, the indices of jobs may be used as attributes to encourage or inhibit assignment of certain jobs to certain positions, hence the method follows certain search directions.

To understand the TS metaheuristic, first consider a general combinatorial optimization problem of the form

$$\begin{aligned} &\text{Minimize} \quad c(x) \\ &\text{subject to} \quad x \in X, \end{aligned}$$

where X is a discrete set of feasible solutions. Traditional local search approaches start from an initial solution $x_0 \in X$. Every feasible solution $x' \in X$ in some neighborhood of x_0, $N(x_0)$, is obtained through slight *perturbations* in x_0 and evaluated as $c(x')$. A *perturbation* or *move* is an atomic change that transforms x_0 to x'. If there does not exist a solution $x' \in N(x_0)$ better than x_0 then the procedure stops. Otherwise, such $x' \in N(x_0)$ is designated as the new x_0 and the procedure iterates similarly.

One obvious drawback of this procedure is that it generates a monotone sequence of improving solutions, until a local optimum (a solution better or no worse than each of its neighbors) is identified. In a sense, the procedure gets stuck in local optima. TS in its most basic form is a local search method with an additional wrinkle that avoids getting stuck in local optima: In each iteration, TS chooses the best move regardless of whether or not it improves the current

solution. If the move is non–improving (and the current solution is at least as good as its parent), then the search process has encountered a local optimum and thus begins the process of escape from it immediately. In addition, TS is characterized by the following key components.

- **Tabu List:** After an iteration in which a non–improving move is accepted, the neighborhood of the new solution contains the previous solution (from which it is generated) that can now be obtained by an improving move. If that move is accepted, the search will be *cycling* back and forth between these two solutions. *Tabu List* stores information that is used to avoid generating solutions visited recently. A move is considered *tabu* and is discarded if it is recorded in the tabu list, or in other words, if its *tabu tenure* has not passed. Tabu tenure is the number of iterations for which the move stays tabu. It is equal to the size of the tabu list that is updated in a cyclic fashion, removing the last move recorded before recording a new move as tabu. Depending on the design, tabu tenure may be fixed (fixed tabu list size) or may change (variable tabu list size) during the search process in order to further influence the search. Tabu list is a short–term (recency–based) and attributive type of memory structure.

- **Aspiration Criterion:** As only limited information is recorded in the tabu list, it is possible that the search process will discard some moves that would generate good solutions. To counteract this potentially negative effect, TS introduces *aspiration criterion* to override the tabu status of a move. The simplest and most widely used form of aspiration criterion is to override the tabu status of a move if it produces a better solution than the best solution found so far in the search process.

- **Candidate List:** In each iteration of the search, TS selects the best non–tabu move among the ones that generate neighbors of the current solution. If the move that is selected is an improving move, then the new solution has the potential of being a local optimum (after the next iteration). These potential local optima are recorded in a list called the *Candidate List.*

- **Index List:** If a solution indeed becomes a local optimum (if it is as good as the solutions in the previous and the next iterations, i.e., if it is as good as its parent and its child solutions) it is recorded in an *Index List.* Candidate and Index lists are explicit memory structures that are used to control the search so that neighborhoods of already explored local optima are not visited again, to monitor when elite solutions are identified, and to identify patterns that are common in the elite solutions.

Note that the neighborhood of a solution x, $N(x)$, is dynamic since it is defined by the move operator that transforms x to $x' \in N(x)$ and some of the moves remain tabu for a certain number of iterations.

In addition to the components presented above, the following features further define the scope of TS.

- **Initial Solution:** As emphasized previously, an initial solution is required to trigger the search. It can be chosen arbitrarily or generated using a favored systematic approach that identifies a potentially good solution.

- **Neighborhood Function:** To move from one solution to another, a neighborhood function that defines the set of new solutions within some neighborhood of a given solution must be designed. Since TS provides a means for traversing the solution space in a clever and economical way, simple moves

that transform a given solution to another can be effectively used to define a neighborhood of a solution.

- **Objective Function Evaluation:** The purpose of the search is to identify a solution (or solutions) that optimize some objective function. Therefore, each time a new solution is generated, its objective function value must be evaluated or the incremental value of the move that results in the solution must be computed.

- **Long–Term Memory Function:** Long term memory components of TS operate primarily as a basis for intensifying and diversifying the search. The fundamental elements of intensification and diversification strategies are already present in the short–term memory component of TS. The tabu list has an intensification role by temporarily locking in certain locally attractive attributes (those belonging to moves recently evaluated to be good), while it also has a diversification role by forcing new choices to introduce attributes that are not among those recently discarded. Long–term memory components can be used to record patterns that are common in elite solutions. Hence, long–term memory functions are usually *frequency–based.* The frequency of certain moves that generated elite solutions or the frequency of occurrence of certain patterns in elite solutions are recorded appropriately. In an intensification strategy, TS seeks new solutions that exhibit these patterns by correspondingly restricting or penalizing available moves, or by crafting a new initial solution that is obtained by putting together common components of elite solutions. Likewise, in a diversification strategy, solutions exhibiting the common patterns may be discouraged by either penalizing certain moves, but also by crafting a new initial solution that

does not exhibit any of the common patterns of the elite solutions found so far. Most often, long–term memory components based on maximal and minimal frequencies are used for intensification and diversification purposes, respectively.

- **Long–Term Memory Function Based on Maximal Frequency (LTM–Max):** The move or pattern that is found *most frequently* in the elite solutions is used as a means for intensification. In production scheduling, for instance, the number of times each job is assigned to each position in the elite sequences may be counted for a certain number of iterations and recorded in a job–position frequency matrix. Then, the job–position pair corresponding to the maximal entry in the matrix can be fixed in the initial solution and the search can *restart* and progress accordingly. Similarly, a new initial solution can be constructed by assigning jobs to positions, starting with the assignment corresponding to the highest frequency, then continuing with the assignment corresponding to the second highest frequency, and so on.

- **Long–Term Memory Function Based on Minimal Frequency (LTM–Min):** In contrast to LTM–Max, the *least frequently* encountered move or pattern may be identified and advantageously used to diversify the search. In the production scheduling example, the job–position pair corresponding to the minimal entry in the matrix can be fixed throughout the search. Alternatively, based on minimal frequencies, the frequency matrix can be used to construct a new starting solution as different as possible from the solutions generated throughout the previous history of the search process.

- **Stopping Criteria:** Since TS will not stop when it encounters a local optimum, other specific stopping criteria must be employed. Most often, the search is terminated when
  - the best solution found so far has not improved for a certain number of iterations, or
  - a certain number of local optima has been identified (in the index list).

Finally, it should be pointed out that the values of several parameters related to the TS components described above need to be determined for the best results in terms of search efficiency and effectiveness. For instance, preferred tabu list sizes lie in intervals related to problem type and size. Similarly, the best values of the parameters such as the limit on the number of iterations without improvement (used to stop the search) or the number of restarts (for intensification/diversification purposes) must be set a priori. Since there is generally not a theoretical clue on the best value of a parameter, preliminary experiments are performed to test different values of the parameters and identify the best ones.

5.2. Components of the Proposed Tabu Search Algorithms

Within the context of group scheduling, the application of tabu search exhibits *two levels* of search. The first level, called the *outside level*, involves the search for a better board group sequence. Given a board group sequence, the second level, called the *inside level*, involves the search for a better individual board type sequence within each group. The final solution is composed of the best group sequence together with its best board type sequence within each group that results in the minimum makespan/mean flow time.

At the outside level an *outside tabu search* algorithm is applied to move from one group sequence to another, and at the inside level an *inside tabu search* algorithm is invoked to move, within each group, from one board type sequence to another. Once a group sequence is identified during the outside search, the search process switches to the inside search in order to find the best board type sequence as well as the resulting makespan/mean flow time of the whole sequence. When the solution corresponding to the best individual board type sequence is identified, the search process switches back to the outside level to search for a newer and better group sequence. As a consequence, the entire search process switches back and forth between the inside and outside levels to finally identify the best solution for the problem.

The next subsections address several issues related to the inside and outside tabu search algorithms.

5.2.1. Initial Solution

The next chapter presents a simple procedure, procedure *Minsetup*, that identifies quality lower bounds on the makespan and the mean flow time for the two–machine problem. The procedure converts the problem to one with sequence–independent setup times by determining possible minimum setup times for each board group. Then, for each measure of performance, a complete sequence composed of a sequence of board groups as well as a sequence of individual board types within each group is constructed. The sequences identified as described are used as the initial sequences to trigger the tabu search algorithms. For three–machine problems, the same algorithms are used to identify initial sequences by disregarding the last machine and considering the first two machines only.

5.2.2. Neighborhood Function

The notation introduced in the previous chapter is still followed in this chapter. For instance, N denotes the number of board groups and n_g the number of individual board types within each group, while σ denotes a given solution (sequence) which is composed of a board group sequence and a sequence of individual board types within each group. Two different neighborhoods of such a feasible solution are defined as follows.

- $NG(\sigma)$=$\{\sigma' : \sigma'$ is a neighbor sequence obtained from σ. The neighbors are obtained by performing pairwise exchanges of any two adjacent groups (i.e., exchange groups g and $g+1$ for $g = 1, \ldots, N-1$) including the cyclic exchange that is performed by exchanging the first and the last groups. The sequence of individual board types within each group remains the same.$\}$

  To illustrate, consider the representative example presented on page 34 and the sequence $\sigma = G_1(G_{11}) - G_2(G_{21} - G_{22}) - G_3(G_{31} - G_{32})$. Then,

$$\begin{aligned} NG(\sigma) = \{&G_2(G_{21} - G_{22}) - G_1(G_{11}) - G_3(G_{31} - G_{32}), \\ &G_1(G_{11}) - G_3(G_{31} - G_{32}) - G_2(G_{21} - G_{22}), \\ &G_3(G_{31} - G_{32}) - G_2(G_{21} - G_{22}) - G_1(G_{11})\}. \end{aligned}$$

- $NB(\sigma)$=$\{\sigma' : \sigma'$ is a neighbor sequence obtained from σ. The neighbors are obtained by keeping the same group sequence but performing pairwise exchanges of any two adjacent board types in the groups (i.e., exchange board types b and $b+1$ in group g for $b = 1, \ldots, n_g - 1$ and $g = 1, \ldots, N$) including the cyclic exchange in each group that is performed by exchanging the first and last board types.$\}$

For example, if $\sigma = G_1(G_{11}) - G_2(G_{21} - G_{22}) - G_3(G_{31} - G_{32})$, then

$$NB(\sigma) = \{G_1(G_{11}) - G_2(G_{22} - G_{21}) - G_3(G_{31} - G_{32}),$$
$$G_1(G_{11}) - G_2(G_{21} - G_{22}) - G_3(G_{32} - G_{31})\}.$$

5.2.3. Evaluation of the Solutions

Each solution encountered during the course of the search must be evaluated. Figure 5.1 presents a pseudo code of an algorithm that computes the the makespan or the mean flow time value of a given sequence σ.

5.2.4. Tabu List

The outside tabu search employs pairwise exchange moves to swap any two groups in, say sequence σ, to generate its neighbors in $NG(\sigma)$. After an iteration, one needs to label the *best move* that swaps a group in position p and a group in position p' as tabu. To do this, a tabu list whose elements are the best moves (or the attributes of the moves; in our case the positions p and p') is defined. If the tabu list is full, the oldest entry is deleted before a new move is recorded as tabu.

It should be noted here that, in the actual implementation, instead of defining a list and updating it in a circular fashion, a tabu tenure matrix is defined with the entries representing the iteration number until which the moves will stay tabu. Remember that a move will stay tabu for a number of iterations that is equal to the tabu tenure (which is equal to the size of the tabu list). For instance, if the search is in iteration 7 and the tabu list size is 4, the move will keep its tabu status until the end of iteration $7 + 4 = 11$. In the actual implementation phase, updating a tabu tenure matrix is more efficient than updating a tabu list.

algorithm compute–objective–value

Inputs: σ: A sequence of board groups and board types within groups.
N: Number of board groups.
n_g: Number of board types within group g, $g = 1, \ldots, N$.
m: Number of machines.
$rt_{g,b,k}$: Run time of board type b of group g on machine k, $g = 1, \ldots, N$; $b = 1, \ldots, n_g$; $k = 1, \ldots, m$.
FS_k^g: Set of indices of feeders on which group g requires a component on machine k, $g = 0, \ldots, N$; $k = 1, \ldots, m$.
$CS_{g,k}^f$: The component required of group g on machine k on feeder f, $g = 0, \ldots, N$; $k = 1, \ldots, m$; $f = 1, \ldots, FS_k^g$.
ST_k: Setup time per feeder on machine k, $k = 1, \ldots, m$.
Output: Objective value of the sequence σ.
Let $Finish[k]$ be the completion time of the current board on machine k, $k = 1, \ldots, m$.

$TFT \leftarrow 0$ ▷ Initialize total flow time to 0
for $k \leftarrow 1$ **to** m **do**
  $Finish[k] \leftarrow 0$ ▷ Initialize the finish times to 0
  $S_{\sigma,k}$ =compute–setup–times$(\sigma, N, k, FS, CS, ST_k)$ ▷ See page 39
end for
for $g \leftarrow 1$ **to** N **do**
  for $k \leftarrow 1$ **to** m **do** $Finish[k] \leftarrow Finish[k] + S_{\sigma,k}[(g)]$ ▷ Add setup time
  for $b \leftarrow 1$ **to** $n_{(g)}$ **do**
    $Finish[1] \leftarrow Finish[1] + rt_{(g),(b),1}$ ▷ Add run time
    for $k \leftarrow 2$ **to** m **do**
      $Finish[k] \leftarrow \max\{Finish[k-1], Finish[k]\} + rt_{(g),(b),k}$
    end for
    $TFT \leftarrow TFT + Finish[m]$ ▷ Update total flow time
  end for
end for
if the objective is minimizing the total flow time **then**
  return TFT ▷ Or return $TFT/\sum_{g=1}^{N} n_g$ for the *mean* flow time
else ▷ If minimizing the makespan
  return $Finish[m]$ ▷ Total completion time on the last machine
end if

Figure 5.1. Algorithm for Computing the Objective Value of a Given Sequence

A similar tabu list (or tabu tenure matrix) is designated for the inside tabu search as well, but this time we keep track of the two board types that are exchanged along with the board group they belong to.

5.2.5. Long–Term Memory

A frequency–based long–term memory (LTM) component is employed in both outside and inside tabu search algorithms. In outside tabu search, an LTM matrix keeps track of the number of times each group is assigned to each position in the best group sequences identified in each iteration. Every time the best neighbor of a group sequence is identified, the LTM matrix is updated. In inside tabu search, a similar LTM matrix is utilized for each group in order to keep track of the frequency of board type–position assignments.

5.2.6. Aspiration Criterion

In both inside and outside searches, the chosen aspiration criterion overrides the tabu status of a move if the resulting solution is better than the best solution found so far.

5.3. Algorithmic Steps of the Proposed Tabu Search Algorithms

After describing the necessary components of tabu search algorithms, the flowcharts of the outside and inside tabu search algorithms are presented and the algorithmic steps involved in them are described still following the notation introduced in the previous chapter.

5.3.1. Outside Tabu Search

Figure 5.2 given on the next page presents the flowchart of the outside tabu search algorithm.

Step 1. Initialization: Identify an initial solution, σ_0, composed of a board group sequence and a sequence of board types within each group as described in section 5.2.1. Perform inside tabu search on σ_0 with an effort to identify the best sequence of individual board types within each group. Let σ^* and σ denote the best solution found so far and the current solution, respectively, and set $\sigma^* = \sigma = \sigma_0$. Let OCL and OIL denote outside candidate and index lists, respectively, and admit the current solution to them, i.e., set OCL = OIL = $\{\sigma\}$. Construct an empty outside long–term memory (OLTM) matrix of size $N \times N$, where N is the number of board groups. For each group g assigned to position p in σ, set OLTM$[g, p]$=1. Let OIT denote the number of iterations without improvement in the best solution during the outside search, and set OIT = 0. Let $obj(\sigma)$ denote the objective value of σ and set the outside aspiration level OAL equal to $obj(\sigma)$. Construct OTL as an empty outside tabu list. Based on preliminary experimentation, the size of fixed OTL is $\lfloor N/2 \rfloor$.

Step 2. Neighborhood: Generate the neighbors of the current solution σ as $NG(\sigma)$. Discard any solution in $NG(\sigma)$ that is in OCL. Perform inside tabu search to evaluate each solution in $NG(\sigma)$.

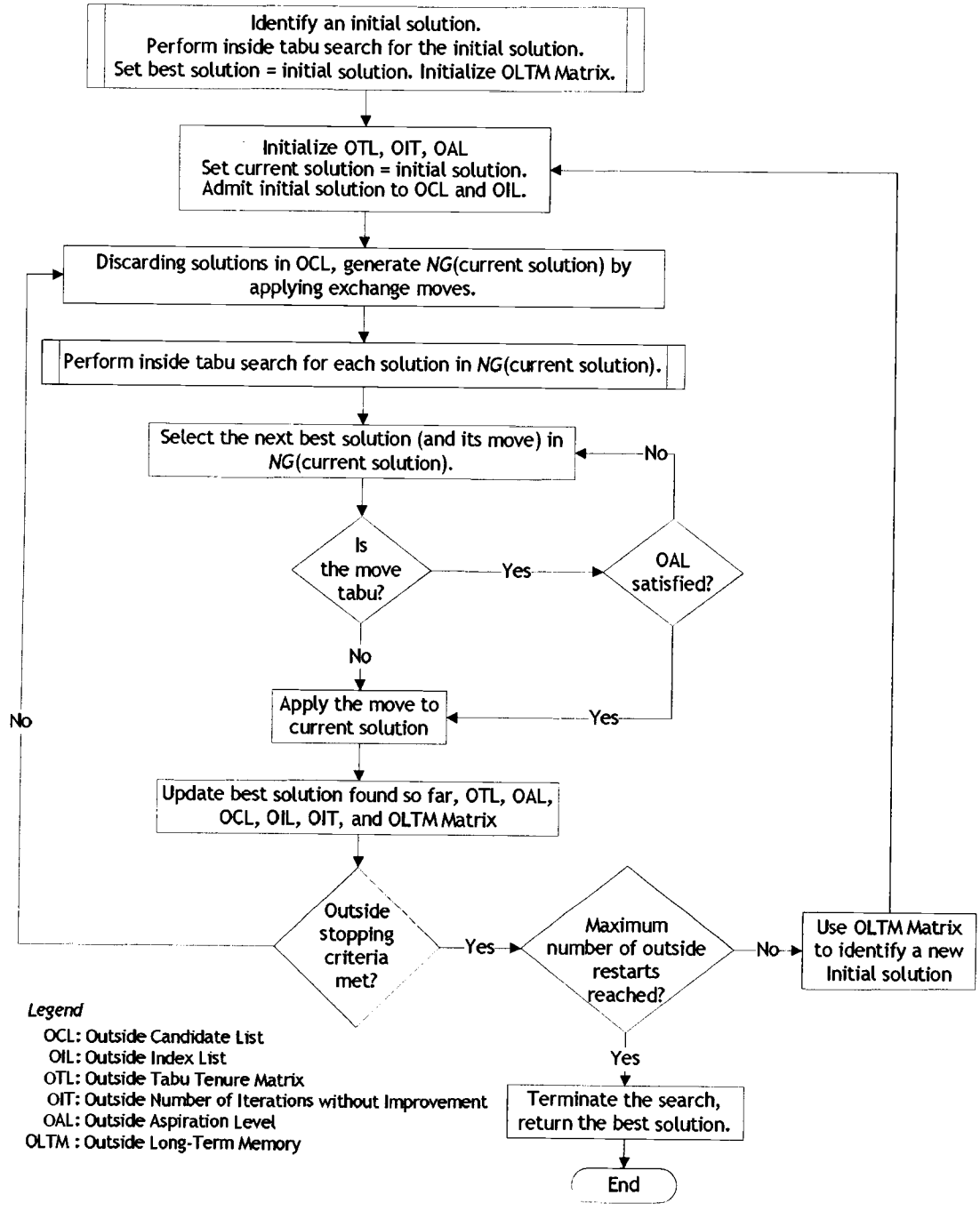

Figure 5.2. The Flowchart of Outside Tabu Search

Step 3. Select Best Move: Select the best solution (and the move that created it) in $NG(\sigma)$. If the move is not tabu, i.e., if it is not in OTL, apply the move to σ (designate the best neighbor as the new σ). If the move is in OTL, check whether the solution satisfies aspiration criterion. Recall that an aspiration criteria that overrides the tabu status of a move if the resulting solution is better than the best solution found so far was chosen. That is, if obj(best neighbor in $NG(\sigma)$) $< obj(\sigma^*)$ apply the move to σ. If the move does not satisfy aspiration criteria, select the next best neighbor and continue similarly. Let the accepted move be the one that swaps the groups in positions p and p'.

Step 4. Update Necessary Components: Admit (p, p') to OTL. Increment OLTM$[g, \bar{p}]$ by 1 for each group g assigned to position $\bar{p}$ in σ. If σ is better than its parent, then it has the potential to become a local optimum, hence append it to OCL. If σ actually is found to be better than its child in the next iteration, it is indeed a local optimum, hence it will be admitted to OIL. If σ is better than the best solution found so far, then set $\sigma^* = \sigma$, OAL $= obj(\sigma)$, and OIT $= 0$, otherwise increment OIT by 1.

At this point, if a variable OTL list size is being used, it is adjusted as follows.

- If the best solution has not been updated for the last $\overline{\text{OIT}}/2$ iterations and if it is the first time that OTL size, |OTL|, is to be changed or was decreased the last time, increase OTL size to 1.5|OTL| (see the next step for explanation of $\overline{\text{OIT}}$). Set OIT $= 0$.

- Similarly, if the best solution has not been updated for the last $\overline{\text{OIT}}/2$ iterations and |OTL| was increased the last time, decrease OTL size to |OTL|/2.

Step 5. Stopping Criteria: Stop if either of the following holds.

- If the best solution has not been updated for the last $\overline{\text{OIT}}$ iterations, i.e., if $\text{OIT} = \overline{\text{OIT}}$. Preliminary experiments suggest that $\overline{\text{OIT}} = \lfloor N/2 \rfloor$.
- If the number of local optima in OIL is equal to $\overline{\text{OIL}} = \lceil N^2/2 \rceil$.

Stop the search if any one of the two stopping criteria is met. If not, go back to Step 2.

Step 6. Restart: When at least one of the stopping criteria is met, necessary adjustments are made to intensify or diversify the search. The OLTM matrix is used to identify a new starting solution and restart the process. However, based on the preliminary experiments, the number of restarts is limited to 2 (start with the very first initial solution and restart two more times after constructing new initial solutions). If the process restarted two times previously, stop the overall search and report the best solution, otherwise continue as follows.

- For intensification purposes, keeping the same sequence of board types within each group as in the best solution found so far, construct a new sequence of board groups using OLTM–Max. To construct such a new sequence from scratch, identify the maximal entry in the OLTM matrix corresponding to the assignment of group g to position $\bar{p}$. Fix group g at position $\bar{p}$, remove g and $\bar{p}$ from further consideration and continue in the same fashion identifying the maximal entry and assigning groups

to positions until all the groups are assigned. Keep the same sequence of board types within groups as in the best solution found so far.

- For diversification purposes, follow exactly the same approach as in intensification, but this time identify entries in the OLTM matrix with the minimal value (i.e., use OLTM–Min).

Finally, designate σ as the new solution just constructed and initialize OTL, OIT, and OAL. Admit σ to OCL and OIL. Go back to Step 2.

5.3.2. Inside Tabu Search

Figure 5.3 presented on the next page exhibits the flowchart of the inside tabu search algorithm, which is similar to the outside tabu search algorithm. However, there are differences as described in the sequel.

Step 1. Initialization: An initial solution σ_0 is provided by the outside tabu search algorithm. Evaluate the objective value of σ_0, $obj(\sigma_0)$, as explained in section 5.2.3. Let σ^* and σ denote the best solution found so far and the current solution, respectively, and set $\sigma^* = \sigma = \sigma_0$. Let ICL and IIL denote inside candidate and index lists, respectively, and admit the current solution to them, i.e., set ICL = IIL = $\{\sigma\}$. For each group g, $g = 1, \ldots, N$, construct an empty inside long–term memory matrix (ILTM$_g$) of size $n_g \times n_g$ where n_g is the number of board types within group g. For each board type b of group g assigned to position $\bar{p}$ in σ, set ILTM$_g[b, \bar{p}]$=1. Let IIT denote the number of iterations without improvement in the best solution during the inside search and set IIT = 0. Set the inside aspiration level IAL equal to

$obj(\sigma)$. Construct ITL as an empty inside tabu list. Based on preliminary experimentation, the size of fixed ITL is $\lfloor \sum_g^N n_g/N \rfloor$.

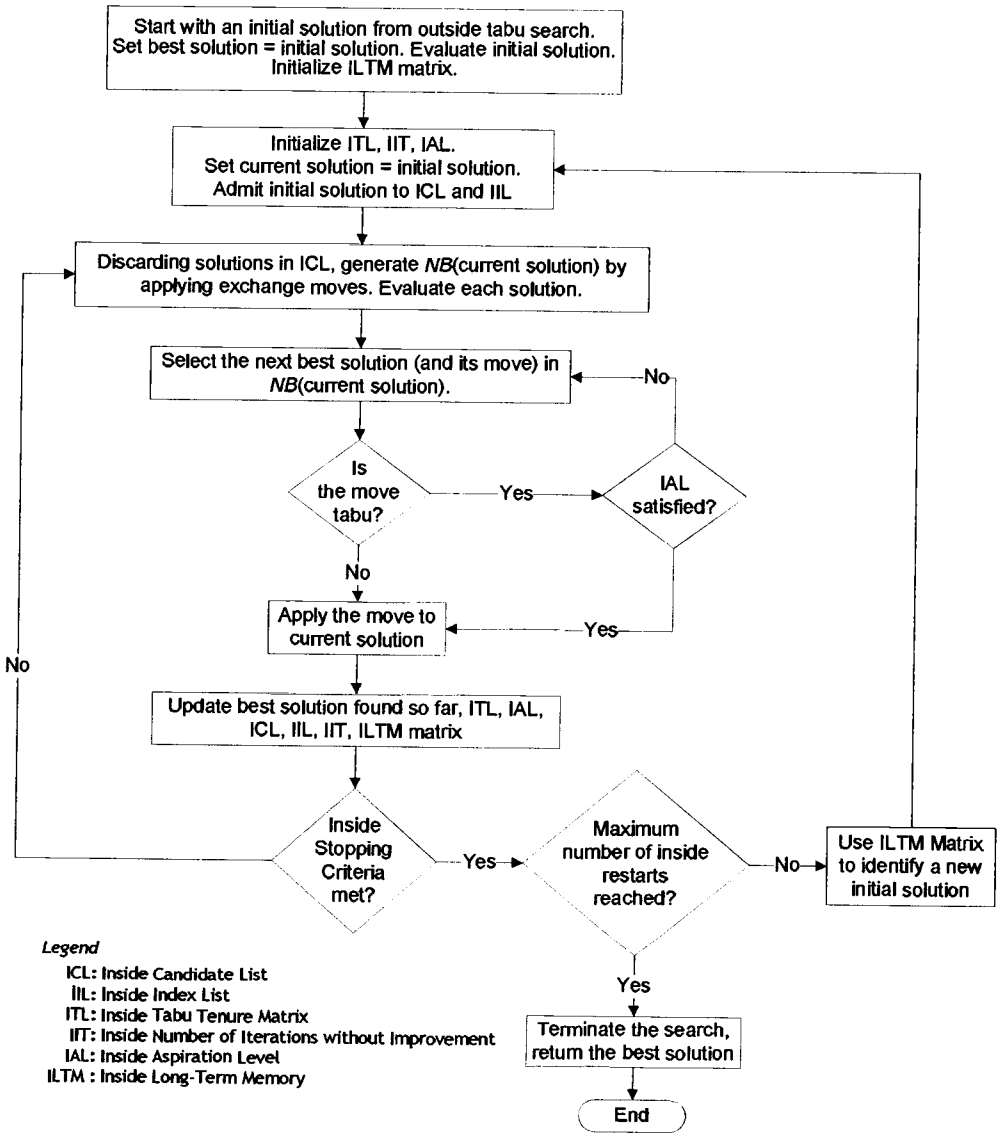

Figure 5.3. The Flowchart of Inside Tabu Search

Step 2. Neighborhood: Generate the neighbors of the current solution σ as $NB(\sigma)$. Discard any solution in $NB(\sigma)$ that is in ICL and evaluate each solution as explained in section 5.2.3.

Step 3. Select Best Move: Select the best solution (and the corresponding move) in $NB(\sigma)$. If the move is not tabu, i.e., if it is not in ITL, apply the move to σ (designate the best neighbor as the new σ). If the move is in ITL, check whether the solution satisfies the aspiration criterion, i.e., if obj(best neighbor in $NB(\sigma)$) $< obj(\sigma^*)$. If the aspiration criterion is not satisfied, select the next best neighbor and its move and continue similarly. Let the selected move be the one that swaps board types in positions p and p' in group g.

Step 4. Update Necessary Components: Admit (g, p, p') to OTL. Increment $\text{ILTM}_g[b, \bar{p}]$ by 1 for each board type b of group g assigned to position $\bar{p}$ in σ. If σ is better than its parent, then append it to ICL since it has the potential to become a local optimum. If σ is actually found to be better than its child in the next iteration, it is indeed a local optimum, hence it will be admitted to IIL. If σ is better than the best solution found so far, then set $\sigma^* = \sigma$, IAL $= obj(\sigma)$, and IIT $= 0$, else increment IIT by 1.

At this point, if a variable ITL size is being used, it is adjusted as follows.

- If IIT $= \overline{\text{IIT}}/2$, i.e., if the best solution has not been updated for the last $\overline{\text{IIT}}/2$ iterations, and if it is the first time that ITL size, |ITL|, is to be changed or was decreased the last time, increase |ITL| to 1.5|ITL| (see the next step for explanation of $\overline{\text{IIT}}$). Set IIT $= 0$.
- Similarly, if the best solution has not been updated for the last $\overline{\text{IIT}}/2$ iterations and |ITL| was increased the last time, decrease |ITL| to |ITL|/2.

Step 5. Stopping Criteria: A similar stoping criteria is employed as in the outside tabu search. Stop if either of the following holds.

- If the best solution has not been updated for the last $\overline{\text{IIT}}$ iterations, i.e., if IIT = $\overline{\text{IIT}}$. Preliminary experiments suggest that $\overline{\text{IIT}} = \lfloor \sum_g^N n_g/N \rfloor$.
- If the number of local optima admitted to IIL is equal to $\overline{\text{IIL}}$ where $\overline{\text{IIL}} = \lfloor nmax \sum_g^N n_g/N \rfloor$.

If both of the two stoping criteria are not met go back to Step 2, otherwise continue with the next step.

Step 6. Restart: Similar to the approach used in the outside tabu search, the ILTM_g matrices (for $g = 1, \ldots, N$) are used to identify a new starting solution to restart the process. However, based on the preliminary experiments, the number of restarts is limited to 2 (start with the very first initial solution and restart two more times after constructing new initial solutions). If the process restarted two times previously, stop the inside tabu search and switch back to the outside search, otherwise continue as follows.

- For intensification, construct a new sequence of board groups as well as a sequence of board types within groups using ILTM_g–Max. To construct such a new sequence, identify the maximal entry in ILTM_g matrix corresponding assignment of board type b in group g to position $\bar{p}$. Fix board type b in group g at position $\bar{p}$, remove board b in group g and $\bar{p}$ from further consideration and continue similarly identifying the maximal entry and assigning board types to positions.

- For diversification purposes, follow exactly the same approach as in intensification, but this time identify entries in $ILTM_g$ with the minimal value (i.e., use $ILTM_g$–Min).

Finally, designate σ as the new solution just constructed and initialize ITL, IIT, and IAL. Admit σ to ICL and IIL. Go back to Step 2.

5.4. Various Tabu Search Algorithms

Several tabu search algorithms are developed to solve our scheduling problems as presented in Table 5.1.

Table 5.1. Features of the Proposed Tabu Search Algorithms

TS Heuristic	Outside Tabu Search Features				Inside Tabu Search Features			
	FTL	VTL	LTM–Max	LTM–Min	FTL	VTL	LTM–Max	LTM–Max
TS1	√				√			
TS2		√				√		
TS3	√		√		√		√	
TS4	√			√	√			√
TS5		√	√			√	√	
TS6		√		√		√		√

Each of the proposed algorithms incorporates a different combination of various features of the tabu search concept. In Table 5.1, FTL stands for *short term memory with fixed tabu list size*, VTL for *short term memory with variable tabu list size*, and LTM–Max (LTM–Min) for *long–term memory based on maximal*

(*minimal*) *frequency.* Note that using LTM–Max means that the algorithm applies intensification, while using LTM–Min results in diversification. TS5, for instance, uses short term memory with variable tabu list size and long–term memory based on maximal frequency (intensification) in both outside and inside levels.

Note also that each of the features has two possible values (exists or not) and FTL and VTL cannot be applied at the same time in the same level. Hence, there are 64 possible combinations. We have considered only 6 of the combinations, which are the most powerful as they are utilizing advanced features in both outside and inside levels, and the effect of intensification (or diversification) is strengthened when it is used simultaneously in both levels. It is not typical using intensification in one level and diversification in the other.

5.5. Demonstration of TS1 for Minimizing the Mean Flow Time

For minimizing the mean flow time, TS1 is demonstrated on the representative example problem presented previously in section 3.1.

Table 5.2. Values of the Parameters of TS1 for the Example Problem

Outside Tabu Search		Inside Tabu Search	
Parameter	Value	Parameter	Value
OTL size	1	ITL size	1
$\overline{\text{OIT}}$	1	$\overline{\text{IIT}}$	1
$\overline{\text{OIL}}$	5	$\overline{\text{IIL}}$	3

Table 5.2 presents the values of the relevant parameters of the outside and inside tabu search algorithms calculated according to the equations given in

section 5.3. Figure 5.4 depicts the tabu structure that shows the available moves and their status (tabu or not) in an iteration of the outside tabu search.

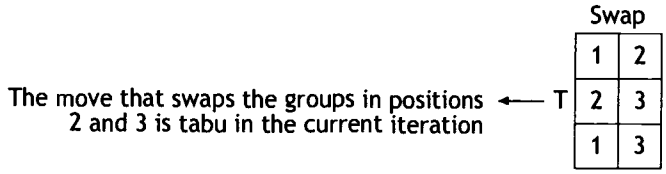

Figure 5.4. Tabu Data Structure for Attributes of Exchange (Swap) Moves

Iteration 0 of the Outside Tabu Search:

Section 6.3.2 on page 100 proposes a method that identifies a lower bound on the mean flow time and a sequence of board groups and board types within each group. The best sequence identified there is $G_2(G_{22} - G_{21}) - G_1(G_{11}) - G_3(G_{32} - G_{31})$ with a lower bound of 3292 seconds. This sequence is used as the initial solution to trigger the search. However, inside tabu search algorithm is applied to this sequence with an effort to identify better board type sequences within groups, while keeping the same sequence of board groups.

Figure 5.5 summarizes the iterations involved in the inside tabu search when applied to the initial solution where MFT denotes the mean flow time. In iteration 0 of the inside tabu search, the initial solution is inserted as the first entry into the ICL and IIL, and IAL is set to MFT of the initial solution which is 4192.8. In iteration 1, the neighborhood of the initial solution is generated using $NB(\cdot)$ and each neighbor is evaluated. The best neighbor is the sequence $G_2(G_{22} - G_{21}) - G_1(G_{11}) - G_3(G_{31} - G_{32})$ with a MFT of 4195.4. Since this solution has an inferior MFT compared to its parent, it is *not* admitted to ICL. At this point, since the number of iterations without improvement reaches $\overline{\text{IIT}} = 1$,

the inside tabu search terminates. In this very small example, the best solution encountered during the inside tabu search is the initial solution.

Iteration 0

Initial Sequence	MFT	ICL	IIL	IAL
$G_2(G_{22}\text{-}G_{21})\text{-}G_1(G_{11})\text{-}G_3(G_{32}\text{-}G_{31})$	4192.8	$G_2(G_{22}\text{-}G_{21})\text{-}G_1(G_{11})\text{-}G_3(G_{32}\text{-}G_{31})$	$G_2(G_{22}\text{-}G_{21})\text{-}G_1(G_{11})\text{-}G_3(G_{32}\text{-}G_{31})$	4192.8

Iteration 1

Neighbors	MFT	ICL	IIL	IAL
$G_2(G_{21}\text{-}G_{22})\text{-}G_1(G_{11})\text{-}G_3(G_{32}\text{-}G_{31})$	4228.4	$G_2(G_{22}\text{-}G_{21})\text{-}G_1(G_{11})\text{-}G_3(G_{32}\text{-}G_{31})$	$G_2(G_{22}\text{-}G_{21})\text{-}G_1(G_{11})\text{-}G_3(G_{32}\text{-}G_{31})$	4192.8
$G_2(G_{22}\text{-}G_{21})\text{-}G_1(G_{11})\text{-}G_3(G_{31}\text{-}G_{32})$	4195.4			

Figure 5.5. Summary of Inside Search in TS1 Applied to the Initial Solution

Now, the search process switches back to the outside level. Figure 5.6 shows a snapshot of the outside tabu search at the end of iteration 0. The best solution just returned by the inside search is inserted into OCL and OIL, and OAL is set to 4192.8. Since there is only one solution in the outside index list, $|\text{OIL}| = 1$.

Initial Solution: $G_2(G_{22}\text{-}G_{21})\text{-}G_1(G_{11})\text{-}G_3(G_{32}\text{-}G_{31})$ MFT = 4192.8

OCL: $G_2(G_{22}\text{-}G_{21})\text{-}G_1(G_{11})\text{-}G_3(G_{32}\text{-}G_{31})$

OIL: $G_2(G_{22}\text{-}G_{21})\text{-}G_1(G_{11})\text{-}G_3(G_{32}\text{-}G_{31})$

OAL = 4192.8

Figure 5.6. Summary of Iteration 0 of Outside Tabu Search in TS1

Iteration 1 of the Outside Tabu Search:

Figure 5.7 below shows a snapshot of the various components of the outside tabu search algorithm at the end of iteration 1. The initial (current) solution in iteration 1 is the best solution identified in iteration 0. The neighbors of the initial solution are generated using $NG(\cdot)$ and inside search is applied to each of

the neighbors in the similar way as illustrated in Figure 5.5. The best of the three neighbors is the sequence $G_1(G_{11}) - G_2(G_{22} - G_{21}) - G_3(G_{31} - G_{32})$ with a MFT of 4120. This solution is inserted in OCL since it is better than its parent, OAL is updated as 4120, and the number of iterations without improvement, OIT, is set to 0 since this solution is better than the best solution found so far. The move that resulted in this sequence is now tabu. The outside tabu search proceeds since OIT$= 0 < 1 = \overline{\text{OIT}}$ and $|\text{OIL}| = 1 < 3 = \overline{\text{OIL}}$.

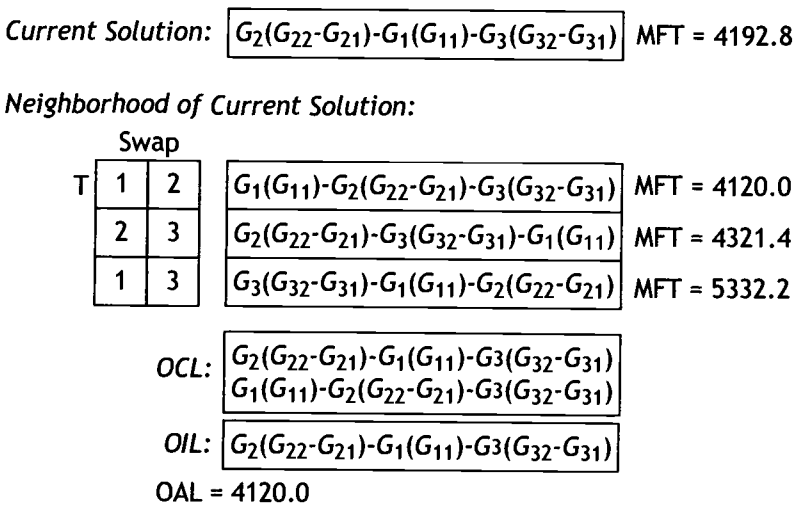

Figure 5.7. Summary of Iteration 1 of Outside Tabu Search in TS1

Iteration 2 of the Outside Tabu Search:

Figure 5.8 shows a snapshot of the various components of the outside tabu search algorithm at the end of iteration 2. The initial (current) solution in iteration 2 is the best solution identified in iteration 1. Note that the move that swaps the groups in positions 1 and 2 now tabu. The best of the three neighbors is the sequence this move created. However, aspiration criterion is not met (the MFT of the corresponding sequence is not better than current OAL which is 4120). Therefore, the next best sequence, which is $G_1(G_{11}) - G_3(G_{32} - G_{31}) - G_2(G_{22} - G_{21})$

with a MFT of 5023.6, is selected. Since this solution is inferior compared to the best solution found so far, OIT is incremented to 1 and the outside tabu search terminates since OIT= 1 = $\overline{\text{OIT}}$. Observe that the best solution obtained in iteration 1 now becomes a local optimum since it is better than its parent and its child; it is admitted to OIL.

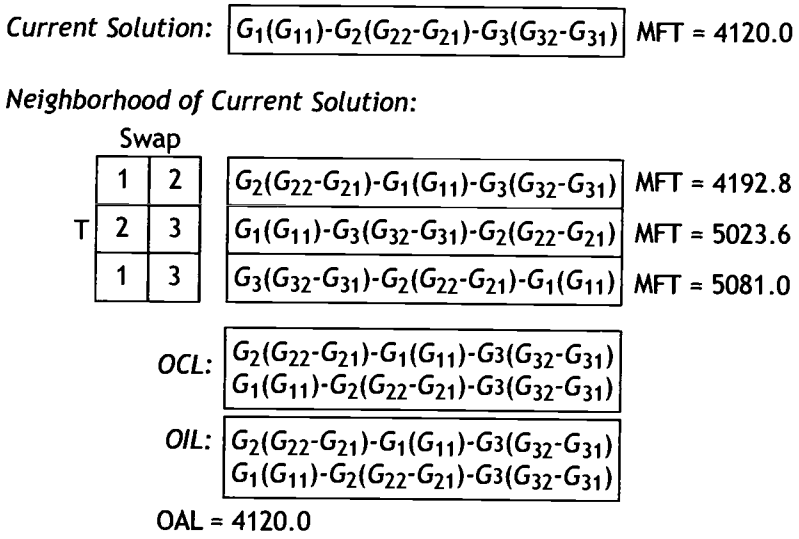

Figure 5.8. Summary of Iteration 2 of Outside Tabu Search in TS1

Finally, Figure 5.9, presented on the next page, summarizes the overall tabu search algorithm. The example problem used to demonstrate the proposed tabu search algorithm establishes the effectiveness of the algorithm. For a group–scheduling problem with N groups, there are $N!$ different group sequences. *Coincidentally*, though, by the end of iteration 2, *all* of the 3! = 6 possible group sequences are identified primarily because the size of the example problem is very small. However, the search, based upon appropriate parameter values for this example problem, did not lead to explicitly investigating every possible board type sequence within each group sequence (for this small size problem instance, this

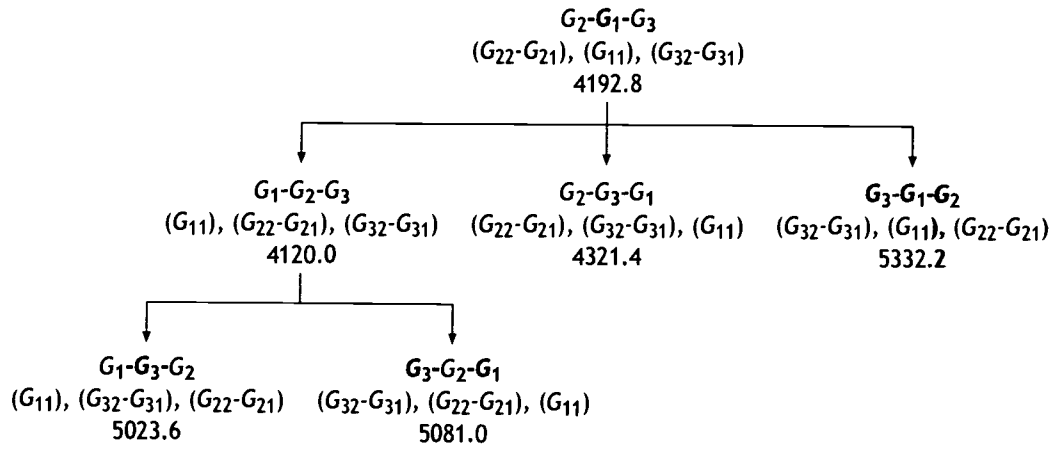

Figure 5.9. Overall Summary of TS1 Applied to the Example Problem

can be explained, in part, by the effectiveness of the initial solution generation mechanism). MILP1 is used to identify the actual optimal solution of this small size problem. The best solution reported by the tabu search algorithm is indeed same as the actual optimal solution. Although based on a single instance, this establishes the premise that the proposed tabu search algorithms are highly effective. In order to comprehensively test the performance of the various algorithms, an extensive experimental study is performed in chapter 8.

6. PROCEDURE MINSETUP

In this chapter, we propose methods that generate lower bounds on the mean flow time and the makespan for the *two–machine* carryover sequence–dependent setup times group–scheduling problem. We present a procedure, which we call "procedure *Minsetup*," as a precursor for identifying effective lower bounds on the makespan and the mean flow time.

Table 3.3 on page 36 in section 3.1, showing the feeder configuration for machine 1 (HSPM) for the representative problem with three board groups, will be used to demonstrate how a lower bound on the total number of feeder changes for each board group can be obtained.

With three groups G_1, G_2, and G_3 to be processed, and R being the reference/initial group, there are six possible group sequences: $R-G_1-G_2-G_3$, $R-G_1-G_3-G_2$, $R-G_2-G_1-G_3$, $R-G_2-G_3-G_1$, $R-G_3-G_1-G_2$, and $R-G_3-G_2-G_1$. If the focus is on G_1 and feeder 3, the *best possible scenario* would require no setup change (as component 101−10 is on feeder 3 of R, in the best scenario, G_1 is scheduled at the first position, right after R; hence no setup is required for G_1 on feeder 3). Likewise, if the focus is on G_1 and feeder 14, the best possible scenario would require no setup change (as component 101−03 is on feeder 14 of G_2, in the best scenario, G_1 is scheduled right after G_2; hence no setup is required for G_1 on feeder 14). On the other hand, if the focus is on feeder 5, the best possible scenario would require one setup change irrespective of the position held by G_1 as component 101−08 is not on feeder 5 of R, G_2, or G_3. Using the same analogy for each feeder on which G_1 requires a component, the total *minimum* number of feeder changes/setups required for G_1 is evaluated to be 5 (0+1+1+0+0+1+1+0+1), irrespective of the position held by G_1. Simi-

larly, the total minimum number of feeder changes required for G_2 and G_3 can be evaluated as 6 (1+1+0+1+0+0+0+1+0+1+0+1) and 6 (1+0+1+1+0+1+1+1), respectively. On machine 2 (MFPM), the total minimum number of feeder changes required for G_1, G_2, and G_3 are 1, 3, and 3, respectively.

For clarity, following the notation introduced in the previous chapters, Figure 6.1 presents the pseudocode of procedure *Minsetup* to compute the minimum setup time for each board group on a given machine.

procedure Minsetup

Inputs: k: The index of the machine.
FS_k^g: Set of indices of feeders on which group g requires a component on machine k, $g = 0, \ldots, N$.
$C_{g,k}^f$: The component required of group g on feeder f of machine k, $g = 0, \ldots, N$; $f = 1, \ldots, FS_k^g$.
Output: Minsetup[g]: The minimum number of setups for group g, $g = 1, \ldots, N$.

```
for g ← 1 to N do
    Minsetup[g] ← |FS_k^g|          ▷ Initialize number of feeder setups of group g
    for all f ∈ FS_k^g do           ▷ For all feeders required of group g
        for g' ← 0 to N and g ≠ g' do   ▷ compare group g with other groups
            if f ∈ FS_k^g' and C_{g,k}^f = C_{g',k}^f then   ▷ If the same component
                Minsetup[g] ← Minsetup[g] − 1   ▷ then no setup on that feeder
                break for loop      ▷ and continue with the next feeder
            end if
        end for
    end for
end for
```

Figure 6.1. Pseudo Code of Procedure Minsetup

The algorithm initially assumes that a feeder setup is needed for each component required of the current board group, resulting in a total number of feeder setups equal to the number of components required of the current board

group. Then, the total number of feeder setups is decreased by 1 for every feeder on which another board group requires the same component. This is repeated for all board groups.

6.1. Lower Bound on the Makespan

Procedure *Minsetup* described above for determining the minimum number of setup changes required of each board group on machine 1 and machine 2 has enabled *converting* a notoriously hard, two–machine sequence–dependent group–scheduling problem with carryover setups into a two–machine sequence–*independent* group–scheduling problem with *no carryovers.* The provision for performing anticipatory setups on machine 2 is yet maintained in the latter (converted) problem structure. Yoshida and Hitomi [122] had proposed a polynomial time algorithm to determine the *optimal* (minimum) makespan for the two–machine group–scheduling problem with sequence–independent anticipatory setups. As the converted problem belongs to this class, the same algorithm can be applied to determine the optimal makespan for the converted problem, which is a lower bound on the makespan of the original problem. This property is stated as Theorem 6.1.1. Then, we present the algorithm that identifies a lower bound on the makespan for the original problem.

Theorem 6.1.1. *The algorithm by Yoshida and Hitomi [122], when applied with the minimum setup times obtained as described in procedure Minsetup above, provides a lower bound on the makespan for the original problem with carryover sequence dependency.*

Proof. Let σ^*_{CS} and $f(\sigma^*_{CS})$ be the optimal (board group and board type) sequence and the optimal makespan, respectively, for the original problem with carryover

sequence dependency. Also let $f'(\sigma^*_{CS})$ be the makespan obtained for the sequence σ^*_{CS} using the minimum setup times computed by procedure *Minsetup* (instead of the actual carryover sequence–dependent setup times). Clearly,

$$f'(\sigma^*_{CS}) \leq f(\sigma^*_{CS}) \tag{6.1.1}$$

Let σ^*_{MS} and $f'(\sigma^*_{MS})$ be the optimal (board group and board type) sequence and the makespan obtained, respectively, by applying the algorithm due to Yoshida and Hitomi [122] with the minimum setup times from the procedure *Minsetup*. Since the algorithm due to Yoshida and Hitomi [122] guaranties an optimal sequence for the converted problem,

$$f'(\sigma^*_{MS}) \leq f'(\sigma^*_{CS}) \tag{6.1.2}$$

From inequalities (6.1.1) and (6.1.2), it follows that $f'(\sigma^*_{MS}) \leq f(\sigma^*_{CS})$, i.e., $f'(\sigma^*_{MS})$ is a lower bound on the makespan of the original problem with carryover sequence dependency. □

The algorithm due to Yoshida and Hitomi [122] presented in Figure 6.2 applied to the converted problem determines a sequence of groups and jobs within groups, which minimizes the makespan of the two–machine group–scheduling problem with sequence–independent anticipatory setup times. Remember that n_g stands for the number of board types within board group g. Also let,

$S_{g,k}$ = Minimum setup time from the procedure *Minsetup* of group g on machine k, $g = 1, \ldots, N$; $k = 1, 2$.

$rt_{g,(b),k}$ = Run time of the *rank ordered* board type b in group g on machine k, $g = 1, \ldots, N$; $b = 1, \ldots, n_g$; $k = 1, 2$.

Algorithm 5.1 – Minimization of the Makespan

1. Use Johnson's rule [69] (see Appendix A) to determine an optimal job (board type) sequence within each group.

2. Use the following approach to determine an optimal group sequence.

   2.1. For each group ($g = 1, \ldots, N$) under the job (board type) sequence determined in step 1, calculate the following values:

$$A_g = S_{g,1} - S_{g,2} + \max_{1 \le r \le n_g} \left\{ \sum_{b=1}^{r} rt_{g,(b),1} - \sum_{b=1}^{r-1} rt_{g,(b),2} \right\}$$

$$B_g = \max_{1 \le r \le n_g} \left\{ \sum_{b=r}^{n_g} rt_{g,(b),2} - \sum_{b=r+1}^{n_g} rt_{g,(b),1} \right\}$$

   2.2. Find the minimum value among the A_g's and the B_g's. Break ties arbitrarily. Let the minimum value correspond to group g'.

   2.3. If the minimum value is equal to $A_{g'}$, place g' at the first available position. If the minimum value is equal to $B_{g'}$, place g' at the last available position.

   2.4. Remove the assigned group, g', from further consideration. If there are no more groups to consider then stop, otherwise go back to step 2.2.

Figure 6.2. Minimization of the Makespan of the Converted Problem

6.2. Lower Bound on the Mean Flow Time

The two–machine sequence–independent setup times group–scheduling problem with the objective of minimizing the mean flow time is $\mathcal{NP}$–hard since a special case of this problem, the two–machine flowshop scheduling problem with the objective of minimizing the mean flow time, is shown to be $\mathcal{NP}$–hard [45]. Therefore, as opposed to the lower bound on the makespan, the lower bound on the mean flow time cannot be evaluated by a polynomial time algorithm maintaining the original *two–machine* structure of the problem (unless $\mathcal{P} = \mathcal{NP}$). However,

Ham et al. [58] propose a polynomial time algorithm guaranteeing an *optimal* mean flow time to the *single–machine* group–scheduling problem with sequence–independent setup times. We take advantage of this fact to evaluate a lower bound on the mean flow time for the original two–machine problem with carryover sequence dependency by constructing two separate single–machine mean flow time minimization problems. When machine 1 is considered, the minimum setup times of groups and run times of jobs (board types) on machine 2 are disregarded, and an optimal (job and group) sequence on machine 1 is identified. Similarly, when machine 2 is considered, the minimum setup times of groups and run times of jobs on machine 1 are disregarded, and an optimal (job and group) sequence on machine 2 is identified. Such minimum mean flow time values evaluated separately for each machine are appropriately revised to finally evaluate a lower bound on the mean flow time of the original problem. Prior to formally stating this procedure, we provide the algorithm by Ham et al [58] for completion. Notice that, as a single–machine problem is now considered, the subscript k is dropped in the steps below. Also let

T_g = The total run time of board group g, i.e., $T_g = \sum_{b=1}^{n_g} rt_{g,b}$, $g = 1, \ldots, N$.

The application of the algorithm presented in Figure 6.3 below, for ordering the jobs in each group and the groups themselves, minimizes the mean flow time of the single–machine group–scheduling problem with sequence–independent setup times (Ham et al. [58]).

Algorithm 5.2 – Minimization of the Mean Flow Time

1. Sequence the jobs within each group according to the shortest run time rule. In other words, sequence the jobs within each group so that

$$rt_{g(1)} \leq rt_{g(2)} \leq \cdots \leq rt_{g(n_g)}$$

2. Sequence the groups so that

$$\left[(S_{(1)} + T_{(1)})/n_{(1)}\right] \leq \left[(S_{(2)} + T_{(2)})/n_{(2)}\right] \leq \cdots \leq \left[(S_{(N)} + T_{(N)})/n_{(N)}\right]$$

Figure 6.3. Minimization of the Mean Flow Time of the Converted Problem

Preliminaries

Let σ^1_{MFT} and $f'(\sigma^1_{MFT})$ be the (board group and board type) sequence and its mean flow time, respectively, obtained on machine 1 by applying steps 1 and 2 above with the minimum setup times from procedure *Minsetup*. Similarly, let σ^2_{MFT} and $f'(\sigma^2_{MFT})$ be the (board group and board type) sequence and its mean flow time, respectively, obtained on machine 2 by applying steps 1 and 2 above with the minimum setup times from procedure *Minsetup*.

Now, let $r_1 = \max\left\{0, \min_g\left\{S_{g,1} + \min_b\{rt_{g,b,1}\}\right\} - S_{(1),2}\right\}$. Note that the setup operation for the first board group on the second machine cannot start before time instant r_1 in *any* sequence. Also let r_2 denote the minimum of the run times of all of the individual board types on machine 2, i.e., $r_2 = \min_{g,b}\{rt_{g,b,2}\}$.

When only machine 1 is considered (i.e., machine 2 disregarded), let the *completion time of the last board type of the last board group* on machine 1 be right shifted by r_2. Suppose that the mean flow time $f'(\sigma^1_{MFT})$ is *reevaluated* as $f'_R(\sigma^1_{MFT})$ with this right–shifted completion time *only for the last board type of the last board group*. Similarly, when machine 2 is the only machine considered (i.e., machine 1 disregarded), let the *start time of the first board type of the first*

board group on machine 2 be right shifted by r_1. Suppose that the mean flow time $f'(\sigma^2_{MFT})$ is *reevaluated* as $f'_R(\sigma^2_{MFT})$ with this right–shift (r_1) added to delay the start time of *every board type in every group* on machine 2. Finally, let $f'_R = \max\left\{f'_R(\sigma^1_{MFT}), f'_R(\sigma^2_{MFT})\right\}$.

Theorem 6.2.1. *f'_R is a lower bound on the mean flow time for the original problem with carryover sequence dependency.*

Proof. Let σ^*_{CS} and $f(\sigma^*_{CS})$ be an optimal (board group and board type) sequence and the optimal mean flow time, respectively, for the the *original* problem with carryover sequence dependency. Also let $f'(\sigma^*_{CS})$ be the mean flow time obtained for the sequence σ^*_{CS} using the minimum setup times provided by procedure *Minsetup* (instead of the actual carryover sequence–dependent setup times). Clearly,

$$f'(\sigma^*_{CS}) \leq f(\sigma^*_{CS}) \tag{6.2.1}$$

When machine 1 is under consideration, the completion time of the last board type of the last board group on machine 2 will at least be equal to the completion time of the last board type of the last board group on machine 1 right shifted by r_2. Likewise, when machine 2 is considered, the start time of every board type in every group on machine 2 will at least be equal to the original start time delayed by r_1.

Clearly, the mean flow time obtained for the job and group sequence with the minimum setup times provided by procedure *Minsetup*, $f'(\sigma^*_{CS})$, must at least be equal to the largest of $f'_R(\sigma^1_{MFT})$ and $f'_R(\sigma^2_{MFT})$. That is,

$$\max\left\{f'_R(\sigma^1_{MFT}), f'_R(\sigma^2_{MFT})\right\} = f'_R \leq f'(\sigma^*_{CS}) \tag{6.2.2}$$

It follows from inequalities (6.2.1) and (6.2.2) that $f'_R \leq f(\sigma^*_{CS})$, i.e., f'_R is a lower bound on the mean flow time of the original problem with carryover sequence dependency. □

6.3. Application of the Procedure Minsetup Based Lower Bounding Methods

This section demonstrates the procedure *Minsetup* based lower bounding methods on the representative example problem presented in section 3.1. The setup time for each board group and run time of each board type on both machines for the converted sequence–independent group–scheduling problem are presented in Table 6.1 below. As previously illustrated in section 3.1, the sequence–independent setup times are evaluated using procedure *Minsetup*, based on the data presented in Tables 3.3 and 3.4, and using the average setup time per feeder on both machines exhibited in Table 3.2.

Table 6.1. Setup and Run Times for the Sequence–Independent Problem

Board Group	G_1	G_2		G_3	
Board Type (job)	G_{11}	G_{21}	G_{22}	G_{31}	G_{32}
Run Time on machine 1 (sec.)	318	319	141	1096	1083
Run Time on machine 2 (sec.)	20	41	3	488	355
Setup time on machine 1 (sec.)	900	1080		1080	
Setup time on machine 2 (sec.)	220	660		660	

6.3.1. Evaluation of the Lower Bound on the Makespan

The application of Step 1 of Algorithm 5.1 to the example problem results in the following board type sequence within each board group: $G_1 : G_{11}$; $G_2 : G_{21} - G_{22}$; and $G_3 : G_{31} - G_{32}$. The application of Step 2 of Algorithm 5.1

to the example problem results in the evaluation of A_g and B_g values for each of the three board groups as shown in Table 6.2.

Table 6.2. Values of A_g and B_g for the Sequence–Independent Problem

Board Group, g	A_g (sec.)	B_g (sec.)
1	998	20
2	839	3
3	2111	335

From Table 6.2, the group sequence is identified as $G_3 - G_1 - G_2$ and the complete sequence comprised of both the board types within each board group and the board groups themselves is $G_3(G_{31} - G_{32}) - G_1(G_{11}) - G_2(G_{21} - G_{22})$ with a makespan of 6020 seconds as shown in the corresponding Gantt chart in Figure 6.4. The value 6020 is a lower bound on the optimal makespan of the original problem with carryover sequence dependency.

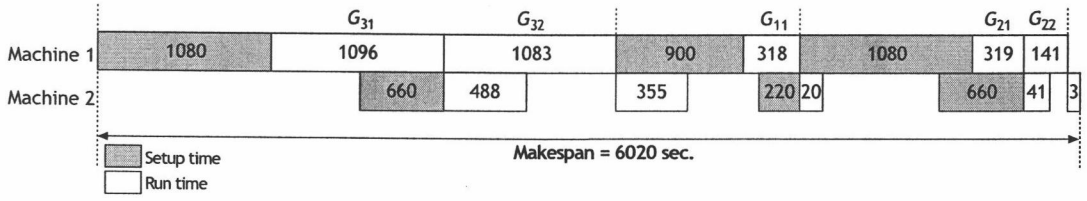

Figure 6.4. Lower Bound on the Makespan of the Example Problem.

6.3.2. Evaluation of the Lower Bound on the Mean Flow Time

Considering machine 1 only, the application of Step 1 of Algorithm 5.2 to the example problem results in the following board type sequences within the

groups: $G_1 : G_{11}$; $G_2 : G_{22} - G_{21}$; and $G_3 : G_{32} - G_{31}$. The application of Step 2 of Algorithm 5.2 results in the following evaluations:

$$\text{For } G_1 : (S_1 + T_1)/n_1 = (900 + 318)/1 \quad = 1218 \text{ sec.}$$

$$\text{For } G_2 : (S_2 + T_2)/n_2 = (1080 + 460)/2 \quad = 770 \text{ sec.}$$

$$\text{For } G_3 : (S_3 + T_3)/n_3 = (1080 + 2179)/2 = 1629.5 \text{ sec.}$$

Thus, the group sequence is identified to be $G_2 - G_1 - G_3$ and the complete sequence comprised of both the board groups and the board types within each group is $G_2(G_{22} - G_{21}) - G_1(G_{11}) - G_3(G_{32} - G_{31})$. The completion times for board types G_{22}, G_{21}, G_{11}, G_{32}, and G_{31} are evaluated as 1221, 1540, 2758, 4921, and 6017, respectively (see Figure 6.5). In addition, $r_2 = \min\{20, 41, 3, 488, 355\} = 3$. Thus, the completion time of G_{31} (the last board type of the last board group) is revised as $6017 + 3 = 6020$. Therefore, when only machine 1 is considered, the revised mean flow time can be evaluated as $(1221+1540+2758+4921+6020)/5 = 3292$ seconds.

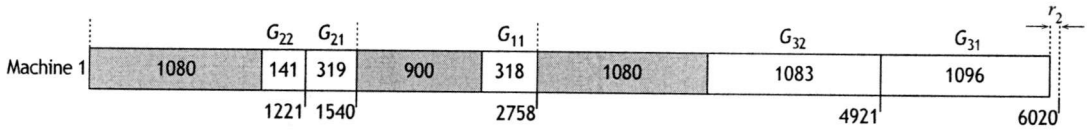

Figure 6.5. Completion Times of the Board Types on Machine 1

A similar approach when used for machine 2, would lead to identifying the sequence $G_1(G_{11}) - G_2(G_{22} - G_{21}) - G_3(G_{32} - G_{31})$ with the corresponding completion times evaluated as 240, 903, 944, 1959, and 2447. Additionally,

$$r_1 = \max\Big\{0,\ \min\{900 + 318, 1080 + 141, 1080 + 1083\} - 220\Big\} = 998.$$

As this right shift (r_1) must be applied to the completion times of *every* board type in each group, the *revised* completion times for board types G_{11}, G_{22},

G_{21}, G_{32}, G_{31} in the above sequence can be evaluated as 1238, 1901, 1942, 2957, and 3445, respectively, as shown in Figure 6.6. Observe that as it is now machine 2 and the setup on it can be performed in anticipation of the arriving board group, the evaluation of r_1 excludes the setup time (220 sec.) required of the first group (which is G_1) in the above sequence. This also impacts the completion time of the rest of the board types. Hence, when only machine 2 is considered, the revised mean flow time can be evaluated as $(1238 + 1901 + 1942 + 2957 + 3445)/5 = 2296.6$ seconds. Finally therefore, the best lower bound on the mean flow time for the original problem is $\max\{3292, 2296.6\} = 3292$ seconds.

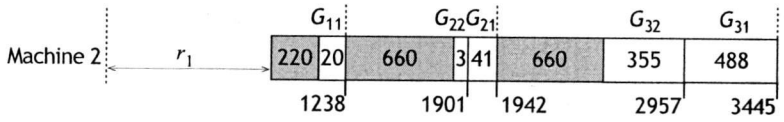

Figure 6.6. Completion Times of the Board Types on Machine 2

The lower bounds identified above are, on the average, close to the actual optimal solutions. For the problem with the objective of minimizing the mean flow time, we perform a computational experiment to quantify the performance of a basic tabu search algorithm with respect to the procedure Minsetup based lower bounds (Logendran et al. [83]). Generally, the lower bounds identified are close to the actual optimal solutions. However, there are individual instances for which the percentage deviation of the tabu search solutions with respect to the lower bounds are not as small, even though the tabu search algorithms identify the actual optimal solutions. For the example problem, the optimal mean flow time is 4120. Its percentage deviation from the lower bound 3292 is 25.15%, which is not acceptable even by industry standards. Additionally, the lower bounding

methods described in this chapter are applicable to two–machine problems. Therefore, the next chapter presents a branch–and–price algorithm, which is capable of identifying tighter lower bounds for both two– and three–machine problems.

7. BRANCH & PRICE ALGORITHM

7.1. Introduction

Column generation is a pricing scheme for large–scale linear programs (LPs). Instead of pricing nonbasic variables by enumeration (as performed in the simplex method), in a column generation approach the most negative reduced cost is identified by solving an optimization problem called the subproblem. Sets of columns are left out of the LP problems because there are too many columns to handle efficiently and most of them will have their associated variable equal to zero in an optimal solution anyway. A subproblem is solved to try to identify new columns to be added to the LP problem.

One of the earliest uses of column generation was by Ford and Fulkerson [41] when they suggested to deal only implicitly with the variables of a multi–commodity flow problem. Dantzig and Wolfe [32] pioneered this fundamental idea, developing a general strategy to extend a linear program columnwise as needed in the solution process, which is now known as the Dantzig–Wolfe Decomposition. First, this approach is discussed to provide some general background on column generation algorithms.

Consider a general LP of the form

$$\begin{aligned} \text{Minimize} \quad & cx \\ \text{subject to} \quad & Ax = b \\ & Dx \leq d \\ & x \geq 0 \end{aligned} \tag{7.1.1}$$

whose constraints are partitioned into a class of global constraints ($Ax = b$) and a class of specific constraints referred to as a subsystem ($Dx \leq d$) where A is an

$m \times n$ matrix and D is an $l \times n$ matrix (and the vector of decision variables x, and the vectors c, b, and d are of appropriate sizes). This formulation is called as the *original* or *compact formulation.*

There are two kinds of situations where the decomposition approach is naturally used. One is when the problem is easy to solve over the subsystem (i.e., $Dx \leq d$ are *nice* constraints because they model a well–known polyhedron, for instance, of a network flow problem), but constraints $Ax = b$ complicate the problem. The second and most important one is when the subsystem is decomposable, i.e., the subsystem $Dx \leq d$ has a block diagonal structure. The global constraints $Ax = b$ are then called *linking* or *coupling* constraints since the blocks would be independent without the presence of them.

Now, let $x^1, x^2, \ldots, x^p$ denote the extreme points of the polyhedron $X = \{x \in \mathbf{R}^{\mathbf{n}}_{+} : Dx \leq d\}$ assuming that X is bounded and non–empty. Any feasible point $x \in X$ can be written as a convex combination of the extreme points of X as $x = \sum_{j=1}^{p} \lambda_j x^j$ where $\sum_{j=1}^{p} \lambda_j = 1$ and $\lambda_j \geq 0$ for $j = 1, 2, \ldots, p$. Applying this variable transformation in problem (7.1.1) results in the following *master problem.*

$$\begin{aligned} \text{Minimize} \quad & \sum_{j=1}^{p} (cx^j)\lambda_j \\ \text{subject to} \quad & \sum_{j=1}^{p} (Ax^j)\lambda_j = b \\ & \sum_{j=1}^{p} \lambda_j = 1 \\ & \lambda_j \geq 0 \qquad j = 1, 2, \ldots, p \end{aligned} \tag{7.1.2}$$

In general, solving the master problem directly is totally impractical because of the enormous number of columns in the formulation. To overcome this difficulty, the master problem is solved using *column generation.* In short, the

idea of column generation is to work with just a subset of the columns, resulting in a *restricted master problem*, and to generate missing columns if and when needed. In a given iteration of the column generation algorithm, one first solves the restricted master problem. Let α be the vector of optimal dual variables associated with the first m rows of master problem (7.1.2) and δ be the optimal dual variable associated with the last row, which is usually referred to as the *convexity constraint*. Then, one can price out new columns by solving the *subproblem*

$$\begin{aligned} \text{Minimize} \quad & (c - \alpha A)x - \delta \\ \text{subject to} \quad & Dx \leq d \\ & x \geq 0 \end{aligned}$$

The optimal solution x^* to the subproblem prices out favorably if $(c - \alpha A)x^* - \delta < 0$ and defines a new column that is added to the master problem, which is solved again. When the subproblem fails to identify a favorable column, the solution to the restricted master problem is optimal to the master problem (7.1.2) *over all possible columns*. It is worth noting that not only the optimal solution, but any solution $\hat{x}$ to the subproblem prices out favorably if $(c - \pi A)\hat{x} - \delta < 0$. Therefore, rather than solving the subproblem to optimality, it is possible to employ an approximation algorithm to solve the subproblem in order to identify new columns with negative reduced costs. The column generation algorithm described above can be generalized to handle multiple subproblems and unbounded convex sets.

Solving the integer problems successfully is based on working with easier problems, called *relaxations*, to obtain bounds approximating the true optimal value. These bounds are used to carry out an *implicit* enumeration of all feasible solutions. This method is called *branch–and–bound*. Conventional branch–and–

bound method uses *LP–relaxation* of the integer programming formulation to provide bounds. LP–relaxation is obtained by relaxing the integrality restrictions on the variables.

Although the decomposition principle was developed for the (continuous) linear programming, it is commonly applied in integer programming. Barnhart et al. [11] and Vanderbeck [127] present several reasons for using formulations with a huge number of variables. First, quality of the bound provided by the relaxation is critical for the efficiency of the branch–and–bound method. A compact formulation of an integer problem generally has a weak LP–relaxation. Frequently, the relaxation can be tightened by a reformulation that involves a huge number of variables. This property is especially important for our purpose of identifying tight lower bounds on the makespan/mean flow time in order to quantify the performance of the proposed tabu search algorithms. Second, a compact formulation of an integer problem may have a symmetric structure that causes branch–and–bound to perform poorly because the problem barely changes after branching. A reformulation with a huge number of variables can eliminate this symmetry. Third, column generation provides a decomposition of the problem into a master and subproblem(s). This decomposition may have a natural interpretation in the contextual setting for the incorporation of additional important constraints. Finally, a formulation with a huge number of variables may be the only choice.

The integration of column generation within branch–and–bound is called *integer programming column generation* or *branch–and–price* (B&P). At a first glance, embedding column generation into branch–and–bound may seem straightforward, but there are fundamental difficulties. For example, conventional branching on variables may not be effective because fixing variables can destroy the

structure of the pricing subproblem. Also, solving the LPs to optimality may not be efficient, in which case different rules will apply for managing the B&P tree.

The decomposition in integer programming involves reformulation of the initial compact integer program as an integer program with many integer variables. In other words, when decomposition is applied to an integer program, an *integer master problem* with a large number of columns is obtained. The LP–relaxation of the master is then solved using column generation. The subproblem is also an integer program. Once an optimal solution to the LP–relaxation of the master problem is identified by column generation, one needs to perform branching on the fractional variables of the master problem. In this context, applying a standard branching rule will not guarantee an optimal solution. After branching, it may be the case that there exists a feasible solution that would price out favorably but this solution is not present in the restricted master problem as a result of the branching decisions. Therefore, in order to find an optimal solution, one must generate new columns after branching. However, suppose that the conventional branching rule of variable dichotomy is used on a variable with a fractional value corresponding to, say, λ_j for some j. In the column generation phase it is possible and highly likely that the optimal solution to the subproblem will be the solution represented by λ_j. Then, it becomes necessary to generate the column with the second best objective value. At depth l of the B&P tree, one may need to generate the column with the l^{th} best solution to the subproblem. Furthermore, even if the l^{th} best solution could be found efficiently, a standard branching rule would not be very effective. Since the master problem has a large number of variables, the B&P tree would be highly unbalanced.

In order to prevent columns that have been eliminated by branching from being regenerated, a branching rule that is compatible with the pricing subproblem

must be chosen. By compatible, we mean that we must be able to modify the subproblem so that columns that are infeasible because of the branching decisions are not generated and the subproblem still remains tractable.

The reader is referred to Vanderbeck [128], Vanderbeck and Wolsey [129], and Barnhart et al. [11] for a detailed description of a generic B&P framework including ways to perform branching and some relevant implementation issues.

B&P has been successfully applied to several archetypical combinatorial optimization problems including but not limited to cutting stock problems (Vanderbeck [127]), vehicle routing problems (Desrochers et al. [34], Desrosiers et al. [35]), crew scheduling problems (Vance et al. [123]), the generalized assignment problem (Savelsbergh [105]), and graph coloring (Mehrotra and Trick [94]). General reviews on the subject include Soumis [111] and Wilhelm [120].

Column generation and B&P has recently been successfully applied to a variety of scheduling problems, as well. Most of these applications address single–machine or parallel–machine scheduling problems. Van den Akker et al. [125] propose a set covering formulation for the single–machine unrestrictive common due date problem. They report that the LP–relaxation of their formulation with asymmetric weights yields integer solutions in all randomly generated problem instances. In another research effort, Van den Akker et al. [126] develop a column generation framework for the time–indexed formulations of machine scheduling problems. They specifically apply it to the single–machine weighted completion time problem. For a parallel–machine scheduling problem with the objective of minimizing the total weighted earliness and tardiness, Chen and Powell [26] propose a B&P algorithm. They formulate a set partitioning master problem and single–machine subproblems. The subproblems are solved via a dynamic programming algorithm guaranteeing optimal solutions. Van den Akker [124] address a

parallel–machine total weighted completion time problem. They formulate the problem as a set covering problem with an exponential number of binary variables, and show that subproblems are solvable in pseudo–polynomial time using a dynamic programming algorithm. In order to construct an initial master problem, they use a randomized solution generator. They run the generator up to 5000 times and collect the best 10 solutions. Chen and Powell [25], Lee and Chen [74], and Chen and Lee [24] also employ column generation algorithms for different parallel–machine scheduling problems.

Bülbül et al. [22] present a decomposition based heuristic approach to a flowshop scheduling problem with the objective of minimizing the sum of tardiness, earliness, and intermediate inventory holding costs. They solve the LP–relaxation of the master problem only approximately using a column generation algorithm. To the best of our knowledge, their paper is the only attempt in published literature to apply column generation to a flowshop scheduling problem.

When decomposition principle is applied to parallel–machine scheduling problems, the subproblems are generally identical. On the other hand, similar to what is experienced by Bülbül et al. [22], the subproblems are quite different from each other in the decomposition of our flowshop group–scheduling problems.

The remainder of this chapter presents a Dantzig–Wolfe decomposition approach for the group–scheduling problems investigated in this dissertation. It has been shown that separate single–machine subproblems can be constructed to generate new columns when needed during the course of the B&P algorithm. Ways to obtain tight lower bounds are shown and alternative approaches for solving the column generation subproblems are discussed. In addition, branching rules, compatible with the column generation method, are developed and incorporated

in the subproblems in an efficient way as well. Finally, important implementation details related to the proposed B&P algorithm are discussed.

7.2. A Branch & Price Algorithm

As noted before, a common approach to solve complex scheduling problems is to decompose the original problem into a set of independent subproblems, which are relatively easy to solve, by relaxing some of the constraints. Therefore, an important issue in developing an effective decomposition method is finding a good set of coupling constraints to relax. The relaxed problem needs to be relatively easy to solve, but it should reasonably approximate the original problem so that tight lower bounds are possible. Typically, there is a tradeoff between these two goals; relaxed problems that are relatively easier imply looser lower bounds, and that provide tighter lower bounds are generally harder to solve.

Note that constraint set (4.1.7) in MILP1 presented on page 42 in section 4.1 is the only constraint set that includes variables associated with more than one machine. Each of the other constraints include variables associated with one of the machines only. Essentially, constraint set (4.1.7) *links* the machines in the sense that a board type starts operation on a machine only after it is completed on the previous machine. Thus, if this constraint set is relaxed (i.e., dualized as a constraint set in the master problem), separate single–machine subproblems can be constructed as illustrated in the sequel.

We formulate the multi–machine carryover sequence–dependent setup times group–scheduling problem as an integer programming problem with an exponential number of variables that represent feasible schedules on each machine. There is an infinite number of such feasible schedules on each individual machine.

However, if the run times and setup times are all integral and there is an upper bound on the makespan/mean flow time, we can assume that the number of possible schedules is finite but exponential. Let the number of all feasible schedules on machine k be denoted by T_k. Each of these schedules is characterized by a set of completion times and the sequence of board types in the schedule. This chapter still utilizes the notation introduced in chapter 3 to present the mathematical programming formulations. Additionally, we introduce the following notation.

t $= 1, \ldots, T_k$ feasible schedules on machine k, $k = 1, \ldots, m$.

λ_t^k $= 1$, if schedule t is selected on machine k, and 0 otherwise, $k = 1, \ldots, m$; $t = 1, \ldots, T_k$.

$x_{g,b,j}^{t,k}$ $= 1$, if board type b of group g is in position j on machine k in schedule t, 0 otherwise, $g = 1, \ldots, N$; $j = 1, \ldots, J$; $k = 1, \ldots, m$; $t = 1, \ldots, T_k$.

$C_{j,k}^t$ $=$ The completion time of the board type assigned to slot j on machine k in schedule t, $j = 1, \ldots, J$; $k = 1, \ldots, m$; $t = 1, \ldots, T_k$.

Observe that λ_t^k are decision variables, which we refer to as the *master variables*, while $x_{g,b,j}^{t,k}$ and $C_{j,k}^t$ are parameters. Following this notation, the integer programming master problem (IMP) can be presented as follows.

IMP:

$$z_{IMP}^* = \text{Min} \sum_{t=1}^{T_m} (\sum_{j=1}^{J} C_{j,m}^t) \lambda_t^m$$

subject to

$$\sum_{t=1}^{T_k} \left(C_{j,k}^t - \sum_{g=1}^{N} \sum_{b=1}^{n_g} rt_{g,b,k} x_{g,b,j}^{t,k} \right) \lambda_t^k - \sum_{t=1}^{T_{k-1}} C_{j,k-1}^t \lambda_t^{k-1} \geq 0$$

$$k = 2, \ldots, m; \quad j = 1, \ldots, J \qquad (7.2.1)$$

$$\sum_{t=1}^{T_k} \lambda_t^k = 1 \qquad k = 1, \ldots, m \qquad (7.2.2)$$

$$\lambda_t^k \in \{0, 1\} \qquad k = 1, \ldots, m; \quad t = 1, \ldots, T_k \qquad (7.2.3)$$

The objective is to select the one schedule with the smallest sum of flow times on the last machine. Constraint set (7.2.1) corresponds to constraint set (4.1.7) in MILP1. It makes sure that the schedules selected on the machines ensure that the operation for a board type starts on a machine only after that board type is completed on the previous machine. The set (7.2.2) of convexity constraints makes sure that we select only one schedule on each machine. Finally, (7.2.3) are the integrality (binary) restrictions on the variables.

IMP is an integer program with an exponential number of binary variables. We would like to solve its LP–relaxation using column generation. Therefore, the integrality restrictions on the binary variables are relaxed, by replacing (7.2.3) in IMP with $\lambda_t^k \geq 0$, for $k = 1, \ldots, m$ and $t = 1, \ldots, T_k$. Below, we present LMP as the LP–relaxation of the integer master problem with the objective of minimizing the total flow time. The variables within parentheses before the constraint sets denote the corresponding dual variables.

Still, it is impractical to identify all of the schedules and append them in the linear programming master problem LMP initially. Hence, we start with only $T_k' \ll T_k$ of those schedules on each machine, resulting in a restricted linear programming master problem, which is denoted by RLMP.

LMP:

$$z^*_{LMP} = \text{Min} \sum_{t=1}^{T_m} (\sum_{j=1}^{J} C^t_{j,m}) \lambda^m_t$$

subject to

$$(\alpha_{j,k}) \quad \sum_{t=1}^{T_k} \left(C^t_{j,k} - \sum_{g=1}^{N} \sum_{b=1}^{n_g} rt_{g,b,k} x^{t,k}_{g,b,j} \right) \lambda^k_t - \sum_{t=1}^{T_{k-1}} C^t_{j,k-1} \lambda^{k-1}_t \geq 0$$

$$k = 2, \ldots, m; \quad j = 1, \ldots, J \qquad (7.2.4)$$

$$(\delta_k) \quad \sum_{t=1}^{T_k} \lambda^k_t = 1 \qquad k = 1, \ldots, m \qquad (7.2.5)$$

$$\lambda^k_t \geq 0 \qquad k = 1, \ldots, m; \quad t = 1, \ldots, T_k \qquad (7.2.6)$$

In order to construct the subproblems that are capable of identifying new schedules with the most negative reduced cost on each machine, the dual of LMP (DLMP) is presented next.

DLMP:

$$z^*_{DLMP} = \text{Max} \sum_{k=1}^{m} \delta_k$$

subject to

$$\sum_{j=1}^{J} -C^t_{j,1} \alpha_{j,2} + \delta_1 \leq 0 \qquad t = 1, \ldots, T_1 \qquad (7.2.7)$$

$$\sum_{j=1}^{J} \left(C^t_{j,k} - \sum_{g=1}^{N} \sum_{b=1}^{n_g} rt_{g,b,k} x^{t,k}_{g,b,j} \right) \alpha_{j,k} - \sum_{j=1}^{J} C^t_{j,k} \alpha_{j,k+1} + \delta_k \leq 0$$

$$k = 2, \ldots, m-1; \quad t = 1, \ldots, T_k \qquad (7.2.8)$$

$$\sum_{j=1}^{J} \left(C^t_{j,m} - \sum_{g=1}^{N} \sum_{b=1}^{n_g} rt_{g,b,m} x^{t,m}_{g,b,j} \right) \alpha_{j,m} + \delta_m \leq \sum_{j=1}^{J} C^t_{j,m} \quad t = 1, \ldots, T_m \qquad (7.2.9)$$

$$\alpha_{j,k} \geq 0 \qquad j = 1, \ldots, J; \quad k = 1, \ldots, m \qquad (7.2.10)$$

$$\delta_k \text{ unrestricted} \qquad k = 1, \ldots, m \qquad (7.2.11)$$

Recall from duality in linear programming that the reduced cost of a primal variable corresponds to the infeasibility in the associated dual constraint. Also observe that constraint set (7.2.7) is associated only with the master variables defined for machine 1, i.e., δ_1 and λ_1^t for $t = 1, \ldots, T_1$, as well as it includes completion time variables defined for machine 1 only. Therefore, the following subproblem needs to be solved in order to identify the schedule with the smallest reduced cost on machine 1.

$$\textbf{SP(1):} \quad z^*_{SP(1)} = \text{Min} \sum_{j=1}^{J} \alpha_{j,2} C_{j,1} - \delta_1$$

subject to

(4.1.2) – (4.1.5)

(4.1.6), (4.1.8) – (4.1.16) for machine 1 only.

Similarly, to identify schedules with the smallest reduced costs on machines 2 through $m-1$, one needs to solve SP(k) for $k = 2, \ldots, m-1$, respectively.

$$\textbf{SP}(k)\textbf{:} \quad z^*_{SP(k)} = \text{Min} \sum_{j=1}^{J} \left((\alpha_{j,k+1} - \alpha_{j,k}) C_{j,k} + \sum_{g=1}^{N} \sum_{b=1}^{n_g} (\alpha_{j,k} rt_{g,b,k}) x_{g,b,j} \right) - \delta_k$$

subject to

$$\sum_{j=1}^{J} C_{j,k} \leq \overline{TFT} \tag{7.2.12}$$

(4.1.2) – (4.1.5)

(4.1.6), (4.1.8) – (4.1.16) for machine k only.

Finally, the subproblem on the last machine has the following form.

$$\textbf{SP}(m): \quad z^*_{SP(m)} = \text{Min} \sum_{j=1}^{J} \left((1 - \alpha_{j,m}) C_{j,m} + \sum_{g=1}^{N} \sum_{b=1}^{n_g} (\alpha_{j,m} rt_{g,b,m}) x_{g,b,j} \right) - \delta_m$$

subject to

$$\sum_{j=1}^{J} C_{j,m} \leq \overline{TFT} \tag{7.2.13}$$

(4.1.2) – (4.1.5)

(4.1.6), (4.1.8) – (4.1.16) for machine m only.

The constraint sets of a subproblem describe a sequence of board groups and board types within the board groups, and evaluate the completion time of each board type so as to minimize the objective function of the subproblem. We actually let the constraints in MILP1 corresponding to a machine *migrate* to the subproblem corresponding to that machine. However, observe that (7.2.12) and (7.2.13) are additionally introduced in the subproblem formulations corresponding to machines 2 through m. The reason for introducing these constraints is to ensure a bounded solution to these subproblems. The completion time variables are defined to be non–negative and from DLMP it is clear that the dual variables $\alpha_{j,k}$ are non–negative. Therefore, all the objective function coefficients of the completion time variables in SP(1) are non–negative. To minimize the objective function of SP(1), all the completion times will be forced to be as small as possible, ensuring a bounded optimal objective function value without further precaution. On the other hand, consider SP(k) for $k = 2, \ldots, m$. At least one of the objective function coefficients of the completion time variables may turn out to be negative. Then, since the subproblems are minimization problems, the completion times with negative coefficients can be increased to infinity pushing the objective function value to negative infinity. Therefore, as a precaution to prevent

unbounded solutions to these subproblems, total flow time is bounded from above. Any solution feasible to the original problem provides an upper bound on the total flow time. If such a solution with total flow time $\overline{TFT}$ is readily available, we can simply consider schedules that are guaranteed to have a total flow time less than or equal to $\overline{TFT}$. Apart from ensuring bounded solutions to the subproblems, one advantage of this approach is that it speeds up the column generation algorithm since we restrict the domain of the completion times of the feasible schedules. Note that new constraint sets (7.2.12) and (7.2.13) are already *valid* for MILP1 and MILP2, and could be included in them. The issue of finding a potentially good initial solution, which also plays an important role in constructing an initial restricted master problem, is discussed later in section 7.7.1.

The simplest form of the column generation algorithm to solve LMP at each node of the B&P tree is presented in Figure 7.1. In order to find an optimal solution to LMP, optimal dual variable values are obtained by solving the restricted LP master problem RLMP. The dual variable values are appended in the subproblem objectives and the subproblems are solved to identify new columns with negative reduced costs. New columns, if identified, are added to RLMP. This procedure is repeated iteratively until the subproblems fail to identify columns with negative reduced costs. At that moment, the optimal solution to RLMP is the optimal solution to LMP over all possible columns.

In the case of minimizing the makespan, a similar decomposition approach is possible and can be described as follows. The objective function of the integer master problem IMP and of its LP–relaxation (LMP) are changed to minimizing the completion time of the last board type over all possible schedules on the last machine, i.e., they are changed to minimizing $\sum_{t=1}^{T_m} C_{J,m}^t \lambda_t^m$. As a result of this change, in DLMP, only the right–hand side of the constraint set corresponding to

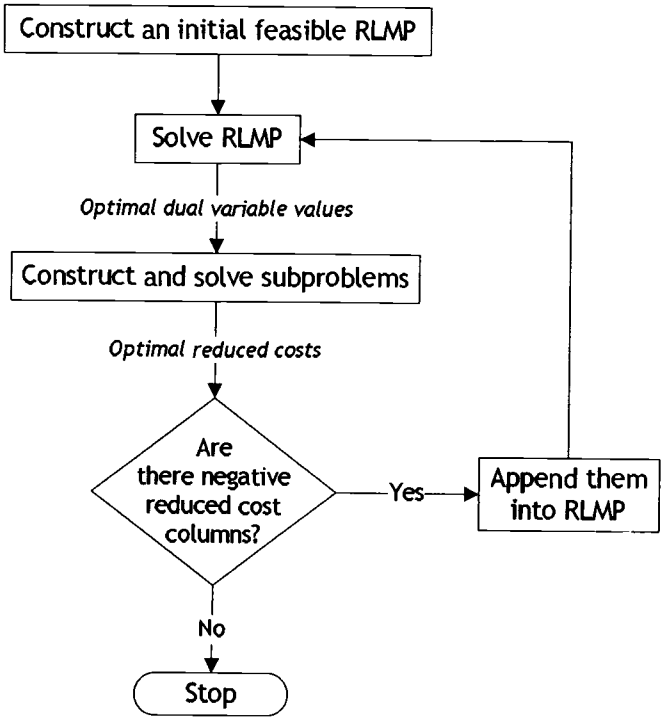

Figure 7.1. The Basic Column Generation Algorithm

the last machine is effected; the right–hand side of constraint set (7.2.9) changes to $C^t_{J,m}$. Consequently, the objective function of the subproblem for the last machine, SP(m), becomes minimizing

$$(1-\alpha_{J,m})C_{J,m} - \sum_{j=1}^{J-1} \alpha_{j,m} C_{j,m} + \sum_{j=1}^{J} \sum_{g=1}^{N} \sum_{b=1}^{n_g} (\alpha_{j,m} rt_{g,b,m}) x_{g,b,j} - \delta_m,$$

while all the remaining subproblems remain unchanged. Unbounded solutions to the subproblems are prevented by introducing $\overline{MS}$ as an upper bound on the makespan. For minimizing the makespan, complete formulations of the integer and LP master problems and the subproblems are presented in Appendix B.

7.3. Lower Bounds

Remember that the primary interest here is in identifying tight lower bounds on the total (mean) flow time/makespan in order to quantify the performance of our tabu search algorithms. The most important reason in choosing a column generation algorithm is the fact that the LP–relaxation of a master problem composed of a large number of columns is tighter than that of a compact formulation. In column generation, lower bounds can be computed not only when the optimality of the LP–relaxation of the master problem is verified, but also in any iteration in which the subproblems are solved to optimality.

Linear programming duality theory tells us that the reduced cost of a primal variable is equal to the infeasibility in the corresponding dual constraint. Therefore, we can compute a lower bound on the optimal objective value of the LP–relaxation of the master problem by constructing a dual feasible solution. In proposition 7.3.1 below, we show how to properly adjust the optimal dual variable values of the restricted LP master problem (RLMP) in any iteration to construct a dual feasible solution (to the *unrestricted* dual problem DLMP). Then, using this dual feasible solution, a lower bound is developed on the optimal objective value, z^*_{IMP}, of the original integer problem.

Proposition 7.3.1. *Given optimal dual variables* $\{\alpha^*_{j,k}, \delta^*_k\}$ *of RLMP in the current iteration,* $\{\alpha^*_{j,k}, \delta^*_k + z^*_{SP(k)}\}$ *is a feasible solution to DLMP* ($j = 1, \ldots, J$ *and* $k = 1, \ldots, m$).

Proof. Remember that the optimal objective value of subproblem SP(k), $z^*_{SP(k)}$, is the infeasibility in the dual constraint (of DLMP) corresponding to machine k. Consider SP(1) and the corresponding constraint set (7.2.7) in DLMP. Adding $z^*_{SP(1)}$ to the left–hand side of constraint set (7.2.7) removes the infeasibility

in (7.2.7). Similarly, adding $z^*_{SP(k)}$ to the left–hand side of constraint set (7.2.8) for $k = 1, \ldots, m-1$ removes the infeasibility in (7.2.8), while adding $z^*_{SP(m)}$ to the left–hand side of the constraint set (7.2.9) removes the infeasibility in (7.2.9). For the optimal dual variables $\{\alpha^*_{j,k}, \delta^*_k\}$ of RLMP in the current iteration, adding $z^*_{SP(k)}$ to δ_k translates into adding $z^*_{SP(k)}$ to the left–hand side of the corresponding constraint set in DLMP. Thus $\{\alpha^*_{j,k}, \delta^*_k + z^*_{SP(k)}\}$ is feasible to DLMP. □

Recall again from linear programming duality theory that when the primal problem is a minimization problem, the objective value of any feasible solution to the dual problem is a lower bound on the optimal objective value of the primal problem. Since $\{\alpha^*_{j,k}, \delta^*_k + z^*_{SP(k)}\}$ is a feasible solution to DLMP (from proposition 7.3.1), its objective value in DLMP is a lower bound on the objective value of LMP, and hence a lower bound on the objective value of the original integer problem. To put it in mathematical terms,

$$\sum_{k=1}^{m}(\delta^*_k + z^*_{SP(k)}) = \sum_{k=1}^{m}\delta^*_k + \sum_{k=1}^{m} z^*_{SP(k)} \leq z^*_{LMP} \leq z^*_{IMP}. \tag{7.3.1}$$

Therefore, $\sum_{k=1}^{m}\delta^*_k + \sum_{k=1}^{m} z^*_{SP(k)}$ is a lower bound on the optimal objective value of IMP when the subproblems are solved to optimality.

Most of the time in the literature, the lower bounds are given as a function of the optimal objective value of RLMP. From the fundamental theorem of duality (see Bazaraa et al. [18]) it is known that the optimal objective values of primal and dual problems are equal. Therefore, at any iteration of column generation, the optimal objective value of RLMP and its dual (call it DRLMP) are equal, i.e.,

$$z^*_{DRLMP} = \sum_{k=1}^{m}\delta^*_k = z^*_{RLMP} \tag{7.3.2}$$

holds. Combining (7.3.1) and (7.3.2) gives

$$z^*_{RLMP} + \sum_{k=1}^{m} z^*_{SP(k)} \leq z^*_{LMP} \leq z^*_{IMP}. \tag{7.3.3}$$

In summary, in any iteration of column generation, the sum of the optimal objective value of the restricted LP master problem and the optimal objective values of the subproblems is a lower bound on the optimal objective value of the original problem.

In addition, note that the optimal makespan and the total flow time to the original problem are both integral since all the setup and run times are integral. Hence, the lower bound obtained above can be improved a bit further by rounding it up to the nearest integer as

$$\left\lceil \sum_{k=1}^{m} \delta^*_k + \sum_{k=1}^{m} z^*_{SP(k)} \right\rceil = \left\lceil z^*_{RLMP} + \sum_{k=1}^{m} z^*_{SP(k)} \right\rceil \leq z^*_{IMP}. \tag{7.3.4}$$

Observe that the left hand side of the above inequality has the largest value when the subproblems are solved to optimality. However, if the subproblems are not solved to optimality but a lower bound on the optimal subproblem objective function value, say $\underline{z}_{SP(k)}$, is obtained, then one can replace $z^*_{SP(k)}$ with $\underline{z}_{SP(k)}$ and still have a valid lower bound on z^*_{IMP}:

$$\left\lceil z^*_{RLMP} + \sum_{k=1}^{m} \underline{z}_{SP(k)} \right\rceil \leq \left\lceil z^*_{RLMP} + \sum_{k=1}^{m} z^*_{SP(k)} \right\rceil \leq z^*_{IMP} \tag{7.3.5}$$

7.4. Branching

The solution to LP–relaxation of the restricted master problem obtained by column generation is not necessarily integral. Once an optimal solution to LMP is identified by column generation, one needs to perform branching on the

fractional master variables. As discussed previously in section 7.1, if a standard branching rule is applied, it may be the case that there exists a feasible solution that would price out favorably but this solution is not present in LMP as a result of the branching decisions. In order to prevent columns that have been eliminated by branching from being regenerated, one must choose a branching rule so that columns that are infeasible because of the branching decisions are not generated. In addition, the branching rule should validly partition the solution space. As noted before, applying a standard branching rule would not be appropriate.

The remedy is to switch back to the original variables (of MILP1) from the master variables. Once the optimal master variable values are obtained, one can compute the values of the original variables as

$$x_{g,b,j} = \sum_{t=1}^{T_k} x_{g,b,j}^{t,k} \lambda_t^k \qquad \forall g,\ b = 1, \ldots, n_g,\ \forall k \tag{7.4.1}$$

Then, we determine the most fractional $x_{g,b,j}$ variable, say as $\hat{x}_{g,b,j}$ corresponding to machine k', and branch on $\hat{x}_{g,b,j}$ variable on machine k'. To do this, we need to make sure that the master variables $\lambda_t^{k'}$ for which $\hat{x}_{g,b,j}^{t,k'} = 1$ are on one branch and the master variables $\lambda_t^{k'}$ for which $\hat{x}_{g,b,j}^{t,k'} = 0$ are on the other branch. In mathematical terms, let $\mathcal{B}$ be the set of schedule indices on machine k' for which $\hat{x}_{g,b,j} = 1$ and $\mathcal{B}^c$ the set of schedule indices on machine k' for which $\hat{x}_{g,b,j} = 0$. That is,

$$\mathcal{B} = \{t \mid \hat{x}_{g,b,j}^{t,k'} = 1, \quad t = 1, \ldots, T_{k'}\}$$

$$\mathcal{B}^c = \{t \mid t \notin \mathcal{B}\} = \{t \mid \hat{x}_{g,b,j}^{t,k'} = 0, \quad t = 1, \ldots, T_{k'}\}.$$

Then, *on the left branch*, we have to include only the master variables for which $\hat{x}_{g,b,j} = 1$, i.e., we must either add

$$\sum_{t \in \mathcal{B}} \lambda_t^{k'} = 1$$

or let the master variables for which $\hat{x}_{g,b,j} = 0$ to 0 as

$$\lambda_t^{k'} = 0, \quad \forall t \in \mathcal{B}^c$$

Similarly, *on the right branch*, we have to include only the master variables for which $\hat{x}_{g,b,j} = 0$, i.e., we must either add

$$\sum_{t \in \mathcal{B}^c} \lambda_t^{k'} = 1$$

or let the master variables for which $\hat{x}_{g,b,j} = 1$ to 0 as

$$\lambda_t^{k'} = 0, \quad \forall t \in \mathcal{B}.$$

After branching, we must make sure that the subproblems at a child node do not generate schedules that do not obey branching restrictions on that child node. Since we are branching on $\hat{x}_{g,b,j}$ on machine k', we simply need to make sure that the subproblem for machine k', SP(k'), does indeed generate schedules for which $\hat{x}_{g,b,j} = 1$ on the left child node. To ensure this, we simply set $\hat{x}_{g,b,j} = 1$ in SP(k'). Similarly, SP(k') should generate schedules for which $\hat{x}_{g,b,j} = 0$ on the right child node. To do this, we simply set $\hat{x}_{g,b,j} = 0$ in SP(k').

Note that, at a node deep down the tree, there may not be columns corresponding to different sequences of board groups and board types within groups. In that case, the branching scheme described above cannot be applied. However, for that node, one can still branch on the completion time variables $C_{j,k'}$. Let j' be the minimum index for which the completion time variables are different among the columns corresponding to machine k'. Also let $\overline{C}_{j',k'}$ be the median of the completion time values in the columns corresponding to machine k'. Then, simply place the columns for which $C_{j',k'} \leq \overline{C}_{j',k'}$ on the left branch and place the columns for which $C_{j',k'} > \overline{C}_{j',k'}$ on the right branch. In the subproblems, the

only modification needed is to add the constraint $C_{j',k'} \leq \overline{C}_{j',k'}$ on the left node, and to add the constraint $C_{j',k'} > \overline{C}_{j',k'}$ on the right node.

7.5. Solving Column Generation Subproblems

Observe in Figure 7.1 that in order to solve the linear programming master problem LMP, the subproblems presented above should be optimized at each iteration of column generation in order to identify columns with the most negative reduced costs. Therefore, if a column with a negative reduced cost exists, the column generation will always identify it. This guarantees that an optimal solution to LMP will be found.

However, as noted in section 7.1, not only the column with the most negative reduced cost, but also *any* column with a negative reduced cost is a candidate to improve RLMP during the course of column generation. Therefore, instead of solving the subproblems to optimality, alternative strategies can be developed.

A heuristic algorithm can be used to approximately solve the subproblems to identify new columns with negative reduced costs, especially if they are computationally too expensive to solve. In that case, one approach is to stop as soon as a negative reduced cost column is identified. This obviously will reduce the computation time per iteration but the overall effect may not be attractive since the number of iterations will probably increase. Another approach is to select all negative reduced cost columns that the heuristic identifies and append them all into RLMP. This approach will require more time per iteration and may decrease the total number of iterations. But since more columns are added to RLMP in each iteration, RLMP grows rapidly and may become harder to solve.

When the subproblems are solved approximately, an optimal solution cannot be guaranteed since there may exist negative reduced cost columns but the heuristic is not able to identify them. Therefore, a two–phase approach must be employed. At the first phase, a fast heuristic is used to solve the subproblems approximately as long as it identifies a negative reduced cost column. In case the heuristic fails to identify such a column, an exact algorithm that solves the subproblems to optimality is invoked to prove optimality or to generate a column with a negative reduced cost. Apart from proving optimality, this stage is also crucial since the lower bounds developed in section 7.3 are valid only when exact optimization is applied.

Our subproblems are single–machine problems with different objective functions. They may require quite some time to solve to optimality. Consider SP(1) for instance. The term $-\delta_k$ is a constant in its objective function. Therefore, SP(1) looks like a single–machine group–scheduling problem with carryover sequence–dependent setup time times with the objective of minimizing the total weighted flow time where $\alpha_{j,1}$ is the weight of the j^{th} job. The single–machine (non–carryover) sequence–dependent setups problem with the objective of minimizing the total flow time (which is represented by $1|s_{g,g'}|\sum C_j$ using the common notation used in the literature to represent scheduling problems) is shown to be $\mathcal{NP}$–hard [101]. Clearly, SP(1) is a generalization of $1|s_{g,g'}|\sum C_j$ and is $\mathcal{NP}$–hard. Likewise, SP(k) for $k = 2, \ldots, m-1$ are special cases of SP(1). Note that with $\alpha_{j,k} = 0$ they look like SP(1). Finally, with $\alpha_{j,m} = 0$, SP(m) reduces to the single–machine carryover sequence–dependent setup times group–scheduling problem with the objective of minimizing the total flow time, which is a generalization of $1|s_{g,g'}|\sum C_j$. In summary, our subproblems are all $\mathcal{NP}$–hard. Therefore, a two–phase approach is employed as shown in Figure 7.2.

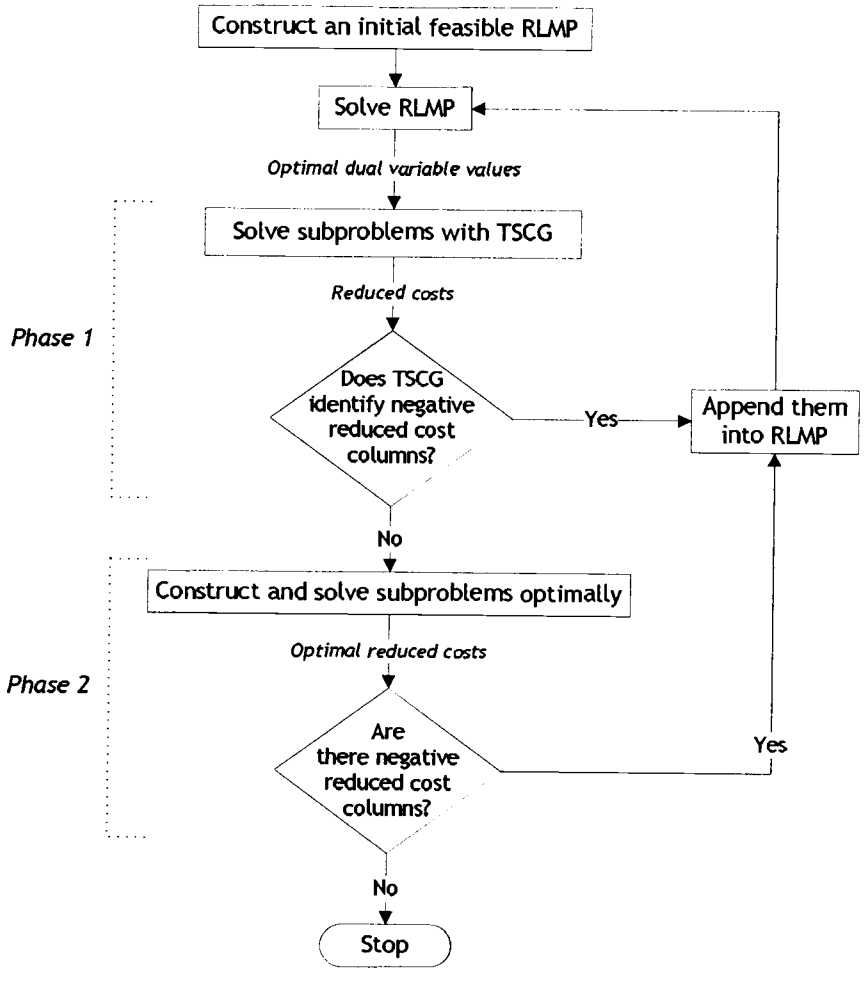

Figure 7.2. The Two–Phase Column Generation Algorithm

We are utilizing our tabu search heuristics but with only short–term memory and fixed tabu–list size (i.e., TS1) to solve the subproblems approximately. To avoid confusion, we call the tabu search heuristic that is used to solve subproblems by TSCG (tabu search column generator). Note that TSCG is different from the tabu search heuristics proposed for the original problem in chapter 5. In an iteration of column generation, we first use TSCG to solve the subproblems. If negative reduced cost columns are identified by TSCG, we append them into

RLMP and continue with the next iteration. In case TSCG fails to identify a column with negative reduced cost, we switch to exact optimization to solve the subproblems.

Remember that in tabu search, we are searching for an optimal/near optimal *sequence* of board groups and board types within each group. Each time the tabu search algorithm encounters a sequence, it should be evaluated. When solving the original problem with tabu search as in chapter 4, the makespan and the mean flow time of a sequence of board groups and board types are evaluated using algorithm compute-objective-value presented in Figure 5.1 on page 74.

A similar algorithm evaluates the objective value of SP(1) for a given sequence. Recall from section 7.2 that in SP(1) boards are always scheduled without any idle time on machine 1 so that the completion times are as small as possible. Therefore, *for a given sequence of board groups and board types within groups*, one can evaluate SP(1) objective easily using algorithm 6.5.1 presented in Figure 7.3.

In order to solve the remaining subproblems using tabu search, however, a convenient method is needed to evaluate the objective values of a given sequence. Since some of the objective function coefficients of the subproblems may be negative, for a given sequence, we need to optimally *schedule* each board type. In other words, we need to *timetable* individual board types to insert *optimal* idle times on the machines (equivalently waiting times for board types) so as to minimize the subproblem objective function. Kanet and Sridharan [72] present a detailed discussion and review of timetabling.

For a given sequence of board groups and board types within each group we actually know the values of the variables $x_{g,b,j}$ that determine the given sequence. We only need to evaluate the completion times of the board types. For fixed values of $x_{g,b,j}$ for the given sequence, we let the formulation evaluate the completion time

Algorithm 6.5.1 – SP(1) objective for a given sequence σ

Inputs: σ: A sequence of board groups and board types within groups
$rt_{g,b,1}$: Run Time of board type b of board group g, $\forall g, b = 1, \ldots, n_g$.
$S_{\sigma,1}[(g)]$: Setup time for the g^{th} group on machine 1 in sequence σ, $\forall g$.
(From algorithm compute-objective-value given on page 3.2)
$\alpha_{j,2}$: Optimal dual variable values from RLMP, $\forall j$.
Output: $z_{SP(1)}$: SP(1) objective value for σ.

$time \leftarrow 0$ ▷ current time on machine 1
$z_{SP(1)} \leftarrow -\delta_1$ ▷ subtract δ_1
for $g \leftarrow 1$ **to** N **do**
  $time \leftarrow time + S_{\sigma,1}[(g)]$ ▷ add setup time
  for $b \leftarrow 1$ **to** $n_{(g)}$ **do**
    $time \leftarrow time + rt_{(g),(b),1}$ ▷ add run time
    $z_{SP(1)} \leftarrow z_{SP(1)} + \alpha_{j,2} \cdot time$ ▷ where $j = b + \sum_{g'=1}^{g-1} n_{(g')}$
  end for
end for

Figure 7.3. Algorithm for Evaluating SP(1) Objective for a Given Sequence

variables which are denoted by $C_{j,k}$ for the j^{th} board type on machine k. We can let the formulation evaluate the carryover sequence–dependent setup times as well, but it is more efficient to simply feed the setup times in the formulation since it is easy to compute them for a given sequence.

In summary, we need to solve the mathematical programming timetabling problem TT(k) in order to evaluate, for a given sequence σ, the objective value of SP(k) for $k = 2, \ldots, m-1$. Note that $S_{\sigma,k}[g]$ is the amount of carryover sequence–dependent setup time required of board group g in sequence σ on machine k, and $\bar{x}_{g,b,j}$ is now a parameter of the form

$$\bar{x}_{g,b,j} = \begin{cases} 1, & \text{if board type } b \text{ of group } g \text{ is at slot (position) } j \text{ in sequence } \sigma \\ 0, & \text{otherwise.} \end{cases}$$

Also let

$$\bar{y}_j = \begin{cases} 1, & \text{if a setup is required at position } j \text{ in sequence } \sigma \\ 0, & \text{otherwise.} \end{cases}$$

TT(k): (for $k = 2, \ldots, m-1$)

$$z^*_{TT(k)} = \text{Min} \sum_{j=1}^{J} (\alpha_{j,k+1} - \alpha_{j,k}) C_{j,k} + \sum_{j=1}^{J} \sum_{g=1}^{N} \sum_{b=1}^{n_g} (\alpha_{j,k} rt_{g,b,k}) \bar{x}_{g,b,j} - \delta_k \qquad (7.5.1)$$

subject to

$$C_{j,k} \geq C_{j-1,k} + \sum_{g=1}^{N} S_{\sigma,k}[g] \sum_{b=1}^{n_g} \bar{x}_{g,b,j} \bar{y}_j + \sum_{g=1}^{N} \sum_{b=1}^{n_g} rt_{g,b,k} \bar{x}_{g,b,j} \quad \forall j,\ C_{0,k} = 0 \qquad (7.5.2)$$

$$\sum_{j=1}^{J} C_{j,k} \leq \overline{TFT} \qquad (7.5.3)$$

$$C_{j,k} \text{ integer} \quad \forall j \qquad (7.5.4)$$

The objective function (7.5.1) corresponds to the objective of SP(k) for $k = 2, \ldots, m-1$. Note that the last two terms are constants. Constraint set (7.5.2) evaluates the optimal completion times of the board types in sequence σ on machine k.

Similarly, in order to evaluate the objective value of a given sequence σ on the last machine, we need to solve TT(m) presented below.

TT(m):

$$z^*_{TT(m)} = \text{Min} \sum_{j=1}^{J}(1-\alpha_{j,m})C_{j,m} + \sum_{j=1}^{J}\sum_{g=1}^{N}\sum_{b=1}^{n_g}(\alpha_{j,m}rt_{g,b,m})\bar{x}_{g,b,j} - \delta_m \qquad (7.5.5)$$

subject to

$$C_{j,m} \geq C_{j-1,m} + \sum_{g=1}^{N} S_{\sigma,m}[g]\sum_{b=1}^{n_g}\bar{x}_{g,b,j}\bar{y}_j + \sum_{g=1}^{N}\sum_{b=1}^{n_g} rt_{g,b,m}\bar{x}_{g,b,j} \quad \forall j,\ C_{0,m}=0 \qquad (7.5.6)$$

$$\sum_{j=1}^{J} C_{j,m} \leq \overline{TFT} \qquad (7.5.7)$$

$$C_{j,m} \text{ integer} \quad \forall j \qquad (7.5.8)$$

Observe again that the objective function (7.5.5) corresponds to the objective of SP(m). Again, the last two terms in (7.5.5) are constants. Constraint set (7.5.6) evaluates the optimal completion times of the board types in sequence σ on the last machine.

The two formulations given above are very small mathematical programs programs, which can be solved extremely efficiently.

Of course, we have to incorporate the branching decisions in our tabu search algorithm so that the columns identified by it obey those decisions. If a solution, which is identified during the course of the tabu search algorithm at a node of the B&P tree, violates at least one of the branching decisions associated with that node, then that solution is discarded.

7.6. Stabilized Column Generation

Column generation is known to be prone to convergence difficulties. Even when solving instances of the same size, the number of iterations can vary by one or

two orders of magnitude. This can be a serious obstacle when implementing a B&P algorithm. While usually a near optimal solution is approached considerably fast, only little progress per iteration is made as the algorithm closes to identifying the optimum solution. Also, it may be relatively time consuming to prove optimality of a degenerate optimal solution. In a matter of speaking, the solution process exhibits a long tail, hence this phenomenon is called the *tailing-off effect.* In order to alleviate this problem, du Merle et al. [38] propose a dual stabilization technique, which is based on a simple observation: the columns that will be part of the final solution are only generated in the very last iterations, when the dual variables are already close to their optimal values. They also observed that dual variables may oscillate erratically in the first iterations, leading to "extreme columns" that have no chance of being in the final solution.

Our approach to stabilize and accelerate the column generation algorithm is similar to the one proposed by du Merle et al. [38] and based on imposing bounds on the dual variable values by introducing *artificial* variables to the master problem. In our case, the benefit of introducing artificial variables is two fold: First, the dual variables are initially "bounded" in some interval, so that the dual variables smoothly converge to their optimal values. Second, the way we bound the dual variables has a very important impact on the objective function coefficients of the subproblems. We constrain the dual variables in such a way that the subproblems become bounded even without the constraint sets 7.2.12 and 7.2.13 which enforce an upper bound on the total flow time. We show in the sequel that when the subproblems are bounded, they become much easier to solve.

Consider introducing $w_{j,k}$ for $j = 1, \ldots, J$ and $k = 1, \ldots, m$ as the artificial variables to constraint set 7.2.1 in the IMP as follows.

IMPA:

$$z^*_{IMPA} = \text{Min} \sum_{t=1}^{T_m} (\sum_{j=1}^{J} C^t_{j,m}) \lambda^m_t + \sum_{j=1}^{J} w_{j,m}$$

subject to

$$(\alpha_{j,k}) \quad \sum_{t=1}^{T_k} \Big(C^t_{j,k} - \sum_{g=1}^{N} \sum_{b=1}^{n_g} rt_{g,b,k} x^{t,k}_{g,b,j} \Big) \lambda^k_t - \sum_{t=1}^{T_{k-1}} C^t_{j,k-1} \lambda^{k-1}_t + w_{j,k} - w_{j,k-1} \geq 0$$

$$k = 2, \ldots, m; \quad j = 1, \ldots, J \tag{7.6.1}$$

$$(\delta_k) \quad \sum_{t=1}^{T_k} \lambda^k_t = 1 \qquad k = 1, \ldots, m \tag{7.6.2}$$

$$w_{j,k} \geq 0 \qquad k = 1, \ldots, m; \quad j = 1, \ldots, J \tag{7.6.3}$$

$$\lambda^k_t \in \{0, 1\} \qquad k = 1, \ldots, m; \quad t = 1, \ldots, T_k \tag{7.6.4}$$

The LP--relaxation of the IMPA is obtained as usual by relaxing the integrality restrictions on the master variables:

LMPA:

$$z^*_{LMPA} = \text{Min} \sum_{t=1}^{T_m} (\sum_{j=1}^{J} C^t_{j,m}) \lambda^m_t + \sum_{j=1}^{J} w_{j,m}$$

subject to

$$(\alpha_{j,k}) \quad \sum_{t=1}^{T_k} \Big(C^t_{j,k} - \sum_{g=1}^{N} \sum_{b=1}^{n_g} rt_{g,b,k} x^{t,k}_{g,b,j} \Big) \lambda^k_t - \sum_{t=1}^{T_{k-1}} C^t_{j,k-1} \lambda^{k-1}_t + w_{j,k} - w_{j,k-1} \geq 0$$

$$k = 2, \ldots, m; \quad j = 1, \ldots, J \tag{7.6.5}$$

$$(\delta_k) \quad \sum_{t=1}^{T_k} \lambda^k_t = 1 \qquad k = 1, \ldots, m \tag{7.6.6}$$

$$w_{j,k} \geq 0 \qquad k = 1, \ldots, m; \quad j = 1, \ldots, J \tag{7.6.7}$$

$$\lambda^k_t \geq 0 \qquad k = 1, \ldots, m; \quad t = 1, \ldots, T_k \tag{7.6.8}$$

Then, the dual of LMPA is given as follows.

DLMPA:

$$z^*_{DLMPA} = \text{Max} \sum_{k=1}^{m} \delta_k$$

subject to

$$\sum_{j=1}^{J} -C^t_{j,1}\alpha_{j,2} + \delta_1 \leq 0 \qquad t = 1, \ldots, T_1 \qquad (7.6.9)$$

$$\sum_{j=1}^{J} \left(C^t_{j,k} - \sum_{g=1}^{N} \sum_{b=1}^{n_g} rt_{g,b,k} x^{t,k}_{g,b,j} \right) \alpha_{j,k} - \sum_{j=1}^{J} C^t_{j,k}\alpha_{j,k+1} + \delta_k \leq 0$$

$$k = 2, \ldots, m-1; \quad t = 1, \ldots, T_k \qquad (7.6.10)$$

$$\sum_{j=1}^{J} \left(C^t_{j,m} - \sum_{g=1}^{N} \sum_{b=1}^{n_g} rt_{g,b,m} x^{t,m}_{g,b,j} \right) \alpha_{j,m} + \delta_m \leq \sum_{j=1}^{J} C^t_{j,m} \quad t = 1, \ldots, T_m \qquad (7.6.11)$$

$$\alpha_{j,k+1} - \alpha_{j,k} \geq 0 \qquad j = 1, \ldots, J; \quad k = 2, \ldots, m-1 \qquad (7.6.12)$$

$$\alpha_{j,m} \leq 1 \qquad j = 1, \ldots, J \qquad (7.6.13)$$

$$\alpha_{j,k} \geq 0 \qquad j = 1, \ldots, J; \quad k = 2, \ldots, m \qquad (7.6.14)$$

δ_k unrestricted $\qquad k = 1, \ldots, m \qquad (7.6.15)$

Note that the first three constraint sets are the same in DLMP and DLMPA but constraint sets 7.6.12 and 7.6.13 in DLMPA, with 7.6.14, now imply that

$$1 \geq \alpha_{j,m} \geq \alpha_{j,m-1} \geq \cdots \geq \alpha_{j,2} \geq 0 \qquad j = 1, \ldots, J \qquad (7.6.16)$$

Similar to the two–phase approach in the simplex algorithm, solving LMP using the artificial variables requires a two–stage approach. First, artificial variables are added to LMP (to form LMPA) and new columns with negative reduced costs are generated until the subproblems fail to do so. However, at that moment, there is no guarantee that *all* the artificial variable values will be zero. Therefore, in the second stage, the constraints imposed on the dual variables in the form of 7.6.16 is gradually relaxed by slightly increasing the objective function coeffi-

cient of $w_{j,m}$ for $j = 1, \ldots, J$ in LMPA by a small amount $\Delta \geq 0$. The column generation algorithm is terminated when all the artificial variable values are zero and new columns with negative reduced costs cannot be identified. Generally, the objective function coefficient of $w_{j,m}$ for $j = 1, \ldots, J$ needs to be increased a number of times in order to set all the artificial variables to zero.

Now, we can discuss the impact of adding the artificial variables to the master problem formulation. Recall from section 7.5 that upper bounds on the makespan/total flow time are introduced in SP(k) for $k = 2, \ldots, m$ to ensure that these subproblems are bounded since they can otherwise be unbounded because the objective function coefficients for some of the completion time variables may be negative. Due to the erratic fluctuation of the dual variable values, this is almost always the case in the first iterations. However, with the introduction of the artificial variables, we make sure that these subproblems are bounded. Observe that, constraint set 7.6.12 makes sure that all the coefficients of the completion time variables in SP(k) for $k = 2, \ldots, m-1$ are now nonnegative. Similarly, constraint set 7.6.13 ensures that all the objective function coefficients of the completion time variables in SP(m) are nonnegative. The proposition given below identifies an important property of the optimal solutions of the subproblems when the subproblems are bounded (bounded because of the nonnegativity of the objective function coefficients of the completion time variables).

Proposition 7.6.1. *When the subproblem SP(k) for $k = 1, \ldots, m$ is bounded (when all the objective function coefficients of the completion time variables are nonnegative), the board types within each group follow the shortest processing time (SPT) order in an optimal solution to SP(k).*

Proof. The proof is presented in Appendix C.

Proposition 7.6.1 is very important when solving the subproblems because since the optimal sequence of the individual board types within each board group is known, the subproblems need only identify the optimal sequence of the board groups. Therefore, when solving the subproblems with TSCG, there is no need for the inside search to identify the best board type sequence within each group. In addition, when the subproblems are not bounded, a timetabling LP problem needs to be solved to evaluate each solution encountered during the course of TSCG. However, when a subproblem is bounded, a sequence can be evaluated for that subproblem with an algorithm that is similar to the one presented for SP(1) in Figure 7.3, which further speeds up the process.

Accordingly, the subproblems are modified to make sure that the board types within each group follow the SPT order. In SP(k) for $k = 1, \ldots, m$, the individual board types within each group are re–indexed so that

$$b \leq b' \iff rt_{g,b,k} \leq rt_{g,b',k} \qquad g = 1, \ldots, N; \quad b, b' = 1, \ldots, n_g.$$

The constraint sets 4.1.2–4.1.5 can be replaced in the subproblems with

$$\sum_{g=1}^{N} x_{g,1,j} = 1 \qquad j = 1, \ldots, J - n_g + 1 \tag{7.6.17}$$

$$\sum_{j=1}^{J-n_g+1} x_{g,1,j} = 1 \qquad g = 1, \ldots, N \tag{7.6.18}$$

$$x_{g,1,j} \leq x_{g,b,j+b-1} \qquad j = 1, \ldots, J - n_g + 1; \quad b = 2, \ldots, n_g \tag{7.6.19}$$

Constraint sets 7.6.17 and 7.6.18 assign the first board type within a group to one of the available positions. Remember that the board types within each group are re–indexed in the SPT order and the first board type ($b = 1$) of group g can be assigned to position $(J - n_g + 1)$ at the latest. Then, constraint set 7.6.19

makes sure that the remaining board types are assigned to the $n_g - 1$ consecutive positions. It has been experienced that with these modifications, the subproblems can solved up to 50 times faster.

In summary, artificial variables are introduced to the master problem formulation and the column generation algorithm continues until new columns with negative reduced costs cannot be identified. During this stage, the subproblems are much easier to solve since the board types within each group follow the SPT order. However, if all the artificial variable values are not equal to zero, the objective function coefficients of the nonzero artificial variables are increased by a small amount and the column generation algorithm continues. This is repeated until all the artificial variables are set to zero. During this stage, however, some of the subproblem can become unbounded (bounded with an upper bound on the makespan/total flow time) and become much harder to solve.

On the other hand, proposition 7.6.2 presented below shows that IMPA is a relaxation of the original problem. This means that even without some of the artificial variable values set to zero, we have a valid lower bound. In this case, if the lower bound found so far is close enough to the solution identified by the TS heuristics, we do not have to worry about setting the artificial variable values to zero.

Proposition 7.6.2. *The optimal solution of IMPA is a lower bound on the optimal solution of the original problem (i.e., $z^*_{IMPA} \leq z^*_{MILP1}$).*

Proof. The proof is presented in Appendix C.

The same stabilization method applies to the case when minimizing the makespan as well. Contrastingly, by introducing artificial variables, we can ensure the subproblems SP(k), $k = 1, \ldots, m-1$ are bounded but we cannot ensure that

the subproblem $SP(m)$ is bounded. However, solving the subproblem $SP(m)$ can generally be avoided as described in section 7.7.4 in the sequel. For the LMPA and DLMPA formulations in the case of minimizing the makespan, refer to Appendix B.

7.7. Implementation Details

The performance of a B&P algorithm can be improved by judicious implementation choices. This section discusses several implementation issues such as constructing an initial feasible restricted LP master problem and column management strategy.

7.7.1. Initial Restricted LP Master Problem

To start the column generation algorithm, an initial restricted LP problem has to be provided. The initial RLMP must be *feasible* to ensure that proper dual information is passed to the pricing subproblems.

One approach to construct an initial feasible RLMP is to generate columns using a column generator such as a randomized heuristic. For instance, Van den Akker et al. [124] randomly generate up to 5000 feasible columns and select the best 10 columns to include in the initial RLMP. Yet another approach to construct an initial RLMP involves first assigning values to dual variables so that they are dual feasible, and then constructing subproblems using these dual variable values, and solving them to generate columns to be included in the initial RLMP.

The former approach is followed here to construct an initial restricted master problem. Remember that the tabu search algorithms designed for the original

problem identify a set of local optima (in the outside index list, OIL). The completion times of the board types in these solutions are evaluated on each machine to construct columns on each machine, and the columns are appended into the initial RLMP. In addition, the upper bound on the total flow time in the subproblems, $\overline{TFT}$, is set to the total flow time of the best solution identified by the tabu search heuristic for the original problem.

7.7.2. Adding Columns

As noted before, a two–phase approach is employed to solve the subproblems. First, TSCG (tabu search column generator) is run to solve each subproblem. TSCG goes after the column with the most negative reduced cost, but during the course of the search it may identify many columns with negative reduced costs. Preliminary experiments indicated that adding all of such columns results in a rapidly growing RLMP, making it harder to solve at each iteration of column generation. Although the number of iterations seem to decrease in general, the total computation time increases. Therefore, for each machine, only the best column that TSCG identifies is selected and appended into RLMP.

It is known that columns satisfying certain conditions can be removed from the master problem. It has been tested whether removing columns from the master problem improves the computation time or not. It is found that the master problem does not grow too much and removing columns does not make a significant difference. Therefore, all the columns are kept in the master problem.

7.7.3. Search Strategy and Termination Criteria

Since the primary interest is in identifying tight lower bounds, we use breadth–first search (also referred to as the width search).

The column generation algorithm at any node is terminated when the subproblems fail to identify new columns with negative reduced costs.

The overall B&P algorithm is terminated if an optimal integer solution is identified or if the percentage deviation of the best solution identified by tabu search for the original problem with respect to the best lower bound in B&P is less than or equal to 5% for large size instances and 3% for small size instances.

7.7.4. Special Cases When Solving the Subproblems

We have observed five important cases in which we need not solve some of the subproblems (neither approximately nor optimally).

Case 1: $\alpha_{j,2} = 0$ for $j = 1, \ldots, J$.

Observe that when all $\alpha_{j,2} = 0$, SP(1) objective function becomes minimizing $-\delta_1$ and the constraint set 7.2.7 in DLMP (7.6.9 in DLMPA) implies that $\delta_1 \leq 0$. Therefore, SP(1) cannot identify a new column with negative reduced cost. Hence, SP(1) can be skipped since it cannot identify any column with negative reduced cost.

Case 2: $\alpha_{j,k} = 0$ for $j = 1, \ldots, J$ and $k = 2, \ldots, m-1$.

In this case, the objective function of SP(k) for $k = 2, \ldots, m-1$ becomes $-\delta_k$ and the constraint set 7.2.8 in DLMP (7.6.10 in DLMPA) implies that $\delta_k \leq 0$. Hence, SP(k) can be skipped.

Case 3: $\alpha_{j,k+1} = \alpha_{j,k} =$ constant for $j = 1, \ldots, J$ and $k = 2, \ldots, m-1$.

In this case, the objective of SP(k), $k = 2, \ldots, m-1$ becomes, minimizing $\sum_{j=1}^{J} \sum_{g=1}^{N} \sum_{b=1}^{n_g} \alpha_{j,k}(rt_{g,b,m})x_{g,b,j} - \delta_k$. The first term is nothing but the sum of the run times of the board types on machine k multiplied by $\alpha_{j,k}$ (which is the same value for all $j = 1, \ldots, J$). From the constraint set 7.2.8 in DLMP (7.6.10 in DLMPA), it is clear that $\sum_{j=1}^{J} \sum_{g=1}^{N} \sum_{b=1}^{n_g} \alpha_{j,k} rt_{g,b,k} x_{g,b,j} - \delta_k \geq 0$. Hence, SP($k$) can be skipped since it cannot identify any column with negative reduced cost.

Case 4: $\alpha_{j,m} = 1$ for $j = 1, \ldots, J$.

SP(m) objective becomes minimizing $\sum_{j=1}^{J} \sum_{g=1}^{N} \sum_{b=1}^{n_g} (rt_{g,b,m})x_{g,b,j} - \delta_m$. The first term is nothing but the sum of the run times of the board types on machine m. From the constraint set 7.2.9 in DLMP (7.6.11 in DLMPA), it is clear that $\sum_{j=1}^{J} \sum_{g=1}^{N} \sum_{b=1}^{n_g} rt_{g,b,m} x_{g,b,j} - \delta_m \geq 0$. Hence, SP($m$) can be skipped since it cannot identify any column with negative reduced cost.

In summary, the ability of the B&P algorithm in identifying tight lower bounds is primarily because of (1) the effectiveness of the approximation algorithms developed for solving the subproblems, (2) the identification of certain cases for which some of the subproblems can be skipped without solving, and (3) the impact of the proposed stabilization methods.

8. COMPUTATIONAL EXPERIMENTS

In this chapter, computational experiments are performed in order to determine the best one among the six proposed tabu search algorithms (i.e., TS1, TS2, TS3, TS4, TS5 and TS6) and asses the quality of the solutions identified by the tabu search algorithms. The quality of a solution identified by a heuristic to a problem instance can be best quantified by its (percentage) deviation from the actual optimal solution to that problem instance. However, as discussed previously, the problems investigated in this dissertation are $\mathcal{NP}$–hard in the strong sense, which imply that exact optimization methods such as the branch–and–bound technique that guarantee identifying the actual optimal solution would require an excessive amount of time and memory, especially for large size problem instances. This is the very reason that we propose tabu search algorithms for solving the scheduling problems effectively in a timely manner. While small size problem instances can be solved optimally with branch–and–bound technique reasonably quickly, it is impractical to solve large size instances optimally. This fact, which is intuitively clear, is proven in our experiments as well; we could not solve large size instances, and even some of the small and medium size instances optimally, as we shall present later in detail. Therefore, we quantify the performance of a tabu search algorithm with respect to the best lower bound identified on the optimal solution as

$$\frac{TSvalue - LB}{LB} \cdot 100\% \tag{8.0.1}$$

where $TSvalue$ is the best solution value identified by the tabu search algorithm and LB is the best lower on the optimal solution value.

In order to test the performance of the proposed tabu search heuristics, small and medium to large size problem instances are generated randomly. Recall

from chapter 1 that we are interested in two– and three–machine flowshop PCB assembly systems. The two–machine assembly system was presented in Figure 3.1. In the three-machine assembly system, a HSPM is followed by two MFPMs. In addition, as mentioned in chapter 1, there are typically two types of PCB manufacturers in industry: A *type 1* manufacturer focuses on producing board types that are *similar* and in high volume during the production period, as in bulk orders for PCBs used in cellular phones and personal computers (high–mix low–volume). A *type 2* manufacturer, on the other hand, focuses on producing board types that are somewhat dissimilar and in low volume during the production period, as in PCBs used in medical equipment and the like (low–mix high–volume). We are curious about the quality of the solutions identified by the proposed tabu search heuristics for scheduling problems experienced by both types of manufacturers. Hence, both types of problems are investigated by generating random problem instances descriptive of each, on both two– and three–machine assembly systems.

In a flowshop manufacturing line, a change in the magnitude of setup times on different machines may effect the performance of a heuristic as well as that of a method that identifies lower bounds. In our flowshop assembly lines, characteristically, the number of components assembled on the boards by HSPM is more than that of MFPM. Recall that the setup time required of a board group is the average setup time to change a single component on the machine multiplied by the total number of feeder/component changes required by that board group. Therefore, the setup times on HSPM is generally much higher compared to the setup times on MFPM. It is worth analyzing how a reduction in the magnitude of the setup times on HSPM (relative to MFPM) affects the performance of the tabu search algorithms. The magnitudes of setup times on a machine can be reduced or increased by reducing or increasing the average setup time per feeder on that

machine. The average setup time per feeder values could be modified to consider a low and high value on both types of machines (HSPM and MFPM). Consider an example presented in Figure 8.1, where two levels of average setup time per feeder are considered for each type of machine.

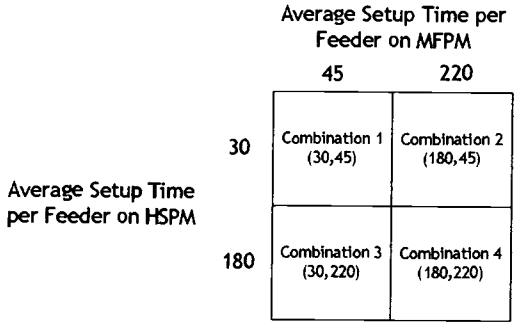

Figure 8.1. An Example with Two Factors Affecting the Setup Times

In the example problem presented in chapter 3, the average setup time per feeder values are 180 seconds and 220 seconds on HSPM and MFPM, respectively. This corresponds to combination 4 in Figure 8.1. For this combination, although the average setup time per feeder on MFPM is greater than that on HSPM, the setup time required of each board group on HSPM is generally greater than that on MFPM. As explained before, this is because the number of components placed on the boards by HSPM is greater than the number of components placed by MFPM, resulting in more feeder setups per board group on HSPM[1]. In summary, even with combination 4, the setup times on HSPM are greater than the setup times on MFPM. Besides combination 4, we are interested in combinations for which the setup times on HSPM are smaller than the setup times on MFPM. Now, consider combination 2 in which the average setup time per feeder on HSPM is

[1]Observe that there are 20 feeders on HSPM while there are 10 feeders on MFPM.

still 180 seconds, but that on MFPM is reduced to 45 seconds. Since the setup times on HSPM were already greater than the setup times on MFPM, reducing the average setup time per feeder on MFPM results in even greater setup times on HSPM than on MFPM. Therefore, this combination does not really have the desired effect. Similarly, combination 1 imitates combination 4, only with reduced average setup time per feeder on both types of machines. Combination 3, however, reduces the average setup time per feeder on HSPM relative to that on MFPM. In this combination, the setup times on HSPM will generally be smaller than the setup times on MFPM, which creates the desired effect. Therefore, there is no compelling reason to consider all four combinations suggested above. Considering combinations 3 and 4 is enough to get the desired effect, which is obtained by two different values of the average setup time per feeder on HSPM, while using the same value on MFPM. In a three–machine assembly line, a reduction in the average setup time per feeder on HSPM will also result in smaller setup times than that on the two following MFPMs, which also is a case of interest to analyze the change in the performance of the proposed tabu search heuristics.

To have a better understanding of the proposed tabu search algorithms and lower bounding methods, problems experimented with are divided into two different sizes. These different sizes define a clear distinction for computational complexity of problems under study. For both types of manufacturers and both two– and three–machine assembly systems, the problem instances with 3 to 8 board groups are referred to as small and medium size instances, while instances with 9 to 14 board groups are referred to as large size instances. This distinction is made mostly because of the ability of MILP3 to solve problems with different sizes. MILP3 could not be used to solve problems with more than 8 board groups due to lack of enough computation memory. This does not imply that all small

and medium size problems can be solved optimally with MILP1 or MILP3 since (1) MILP1 solves a problem instance much slower compared to MILP3, and (2) as the number of board types within each group increases, even MILP3 cannot be solved in reasonable amounts of time.

Another research question of interest is the performance of the proposed lower bounding methods with respect to the actual optimal values. The quality of a lower bound is quantified as its percentage deviation from the actual optimal value, as

$$\frac{|LB - OPT|}{OPT} \cdot 100\% \tag{8.0.2}$$

where OPT is the actual optimal solution value and $|x|$ is the absolute value of x. However, the analysis is *restricted* to the problems for which the actual optimal solutions can be identified. For such problems, the performance of the proposed tabu search heuristics are also investigated by quantifying the quality of heuristic solutions with respect to the actual optimal solutions.

In summary, for minimizing the mean (total) flow time and the makespan, the *primary objectives* of this chapter are as follows.

1. To determine whether the six tabu search algorithms are statistically different in the quality of the solutions they identify. If they are not the same, to determine which of the six tabu search algorithms is (are) the best.

2. To quantify the performance of the tabu search algorithms with respect to the proposed lower bounds.

3. To quantify the performance of the proposed tabu search heuristics with respect to the actual optimal solutions (for the problems for which the actual optimal solution can be identified).

4. To quantify the performance of the proposed lower bounding methods with respect to the actual optimal solutions (for the problems for which the actual optimal solution can be identified).

In addition, the following objectives, which involve assessing the robustness of the proposed tabu search heuristics, are to be addressed by the computational experiments.

5. To assess the effect of the number of machines, size of the problems, and problems for the two different types of electronics manufacturers on the performance of the proposed tabu search heuristics.

6. To determine the effect of reducing the magnitudes of setup times on HSPM (relative to MFPM) on the performance of the proposed tabu search heuristics.

8.1. Data Generation

The data involving a particular instance of the problem is characterized by the following parameters:

- Type of the manufacturer (type 1 or type 2)
- Number of machines
- Number of board groups and board types in each group
- Number of units of each board type to produce in a production period
- Number of components belonging to each board type, the feeder locations of these components, and the number of common components between at least two board types, especially if the board types are in the same group

- Time required to assemble each component on a board type
- Setup time required to change components on the feeders

The number of machines is either 2 or 3, while the number of groups is an input to the problem generator. For type 2 manufacturers, as the volume is low and board types are dissimilar, it is customary to assign a board type to form its own group, and if some similarities existed no more than two board types might be assigned to some board groups. Contrastingly, as similarities are plentiful among the different boards considered for production in the first category (type 1), group technology principles can be conveniently applied to identify board groups that may in fact have more than two board types in each group. Note that the primary focus is to quantify the effectiveness of the proposed heuristics measured by the percentage deviation from the proposed lower bound, as it applies to each type of problems. As such, the problem instances generated in the first type involve board groups with three or more board types, while the second involves board groups with one or two board types. To make the first type of instances manageable, we impose a limit on the maximum number of board types in a group to 5, thus resulting in groups with three, four, or five board types. For both type 1 and type 2 problems, the way the number of board types within each group, number of units of each board type to produce, component–feeder assignments, and run times of board types are generated can be described as follows.

The run time for a unit of a board type is the total time required to assemble all of its components. Since both machines have feeders as component holders and the assembly process is automated, we assume that the time to locate a particular component on the board held on a machine is constant in a board type, and it is the same for all board types in the same group. However, the time

required to assemble the same component is different for board types in different board groups.

The number of board types in each group is determined randomly depending on whether the instance belongs to the first or the second type. For each group of a type 1 instance, a random number, p, is selected from $U[0,1]$. Then the number of board types is set to 3, 4, or 5 if $p < 1/3$, $1/3 \leq p < 2/3$, or $p \geq 2/3$, respectively. But at most $\lceil N/3 \rceil$ groups are allowed to have the same number of board types. For example, if there were seven groups, it would be unrealistic to assign three board types each to more than three groups. For type 2 instances, a random number, p, is generated from $U[0,1]$, and the number of board types of one is assumed if $p < 0.70$, and two if $p \geq 0.70$, as there are more dissimilarities among the boards in this type than type 1. The number of units of each board type is randomly generated as an integer from $U[3,15]$. We assume there is a pool of 125 components (indexed 101-1 through 101-125). The first 75 of these components are assembled on HSPM, and the last 50 components on MFPM. For each component in each group, a random run time (assembly time) is generated from $U[5,20]$. The number of components in each board type is also generated randomly from $U[5,12]$ for HSPM, and from $U[1,5]$ for MFPM. The justification is that HSPM has more feeders than MFPM (20 versus 10), and therefore, more components would be assembled on HSPM.

The assignment of components to individual board types required a lot more thought for generating random problem instances that would emulate the characteristics of real problems in industry. We begin with a randomly selected group, and find the *minimum* number of components required by the board types in that group. Since board types in a group should have a substantial number of components in common, the number of components common among the board

types is assumed to be 80% of this minimum. For instance, if there are three board types with 7, 10, and 8 components, $\lceil 7 \times 80\% \rceil = 6$ components will be common to all three board types. Each of these 6 components are selected randomly from $U[1, 75]$ for HSPM, and from $U[76, 125]$ for MFPM as described before. The component is then assigned to a feeder selected randomly from $U[1, 20]$ on HSPM, and from $U[1, 10]$ on MFPM. As a feeder can only have one component for a board type, if it is identified to hold a particular component, it is marked to indicate that it will not be allowed to hold a different component for the same board type at a later time. Similarly, a particular component for a board type can only be located on one feeder. Thus, if a component is selected for a board type, it cannot be selected for the same board type at a later time. However, this component can be assigned to another board type of another group on the same or different feeder.

Once a component and a feeder are identified for the current group, one of the remaining groups as well as a board type in that group are randomly selected. A random number, p, is then selected from $U[0, 1]$ and if $p < 0.2$, the component under consideration is also assigned to the same feeder for this board type. The same procedure is repeated for the other board types and other groups in a random order. The reason is that some of the components on feeders can be same among the board types in different groups. After such common components are assigned to board types in each group, the number of components is updated to find the additional number of components required by each board type (since now some of the components are already assigned to other board types also). Again, in a random order of groups and board types in each group, components that are not already assigned to a board type are generated randomly and assigned to feeders selected randomly. Once a component is assigned to a feeder for a board type, the

same component is considered for other board types in the same group, as well as for board types in other groups. For example, if there are three board types in a group only two of these board types may have common components on the same feeders. Also, a component of a board type can be the same component of another board type in a different group. At this point, a board type in the same group is identified randomly. A random number, p, is selected from U[0,1], and if $p < 0.5$ the component is also assigned to the same feeder for this board type. The same procedure is then repeated for the other board types in the same group. In doing so, we ensure that a component is not assigned to all of the board types in the same group, as such common component assignments have already been performed as noted above. Similarly, one of the other groups is randomly selected, and a board type, whose selection of the number of components is yet not fulfilled, is identified randomly. A random number, p, is selected from $U[0, 1]$, and if $p < 0.2$, the component is also assigned to the same feeder for this board type. The same procedure is repeated for the other board types.

Following the completion of component assignments, the run times of board types are evaluated. The run time for a board type is simply the *total placement time of its components multiplied by the number of units of that board type* to be produced in the production period. The setup time to change a component on a feeder is assumed constant, and is equal to the average setup time per feeder. Note that, different average setup time per feeder values are used when testing the robustness of the proposed tabu search heuristics against different magnitudes of setup times on the machines.

8.2. Design of Experiment

A common issue in operations research is performing computational experiments in order to compare several algorithms to determine which performs best and what factors affect the performance. The essential role of statistical experimental design, along with general guidelines for computational experiments to compare different algorithms has been discussed by Barr et al. [12], Coffin and Saltzman [27], Greenberg [46], and Hooker [64, 65]. However, there does not seem to be a generally accepted approach in the operations research community to the design and analysis of such computational experiments; only a handful of papers in the literature rigorously use formal methods (Saltzman [27]).

Lin and Rardin [79] use a traditional factorial design in which the response variable is influenced by two factors, namely the algorithm type and the problem type. A number of problems are generated randomly for each combination of algorithm and problem type, where the problem type is specified by setting parameters in the problem generator. The authors recommend *blocking on problem* for comparing different algorithms. Since the problem instances are generated randomly, the instances themselves can cause significant variation in the performance of the algorithms. Therefore, each problem instance is regarded as a "block" and solved by all of the algorithms, removing one source of variability that may obscure the relative performance of the algorithms.

To compare central processing unit (CPU) times of three network reoptimization algorithms and determine the effects of several factors on the CPU times, Amini and Barr [7] employ blocking on problem in the context of a *split–plot experimental design.* The underlying principle in a split–plot design is that *whole plots*, to which levels of one or more factors are applied, are divided into *subplots*

or split–plots each treated by levels of one or more additional factors (see Montgomery et al. [95] for details). Such an experimental design reduces the number of observations required and provides more precise information on the subplot factors than on the whole plot factors. Hence, it is the general practice to assign the factors of most interest to the subplots. In Amini and Barr [7], whole plot factors are the problem class and problem size, while subplot factors are the type of change made to a problem before reoptimization, percentage of the change applied, and the type of the algorithm. Amini and Racer [8], Logendran and Subur [88], and Nance and Moose [96] also employ split–plot designs for comparing alternative algorithms.

To meet the research objectives discussed previously, a split–plot experimental design is employed in this dissertation as well. The response is the percentage deviation from the proposed lower bounds and there are five design factors considered in the experiment. These factors and their levels are presented in Table 8.1, where ASTH stands for average setup time per feeder on HSPM.

Table 8.1. Factors/Treatments and their Levels in the Design of Experiment

Factor Name	Levels
Problem Type (PT)	Type 1, Type 2 ($i = 1, 2$)
Problem Size (PS)	Small/Medium, Large ($j = 1, 2$)
Number of Machines (NM)	2, 3 ($k = 1, 2$)
Avg. setup time per feeder on HSPM (ASTH)	30, 180 ($p = 1, 2$)
Tabu search heuristic (TS)	TS1, TS2, TS3, TS4, TS5, TS6 ($q = 1, \ldots, 6$)

From this point forward, a *problem data* refers to a feeder–component configuration required of all board types and board groups on all machines and the run times of all board types on all machines. *Given a problem type, problem size, and number of machines*, a problem data is generated by the problem generator described in the previous section. A problem data is treated with the combinations of the average setup time per feeder on HSPM (ASTH) factor and the tabu search heuristic factor. Consequently, combinations of problem type, problem size, and number of machines constitute eight whole plots, within which combinations of ASTH and TS heuristics form the subplots, as depicted in Figure 8.2.

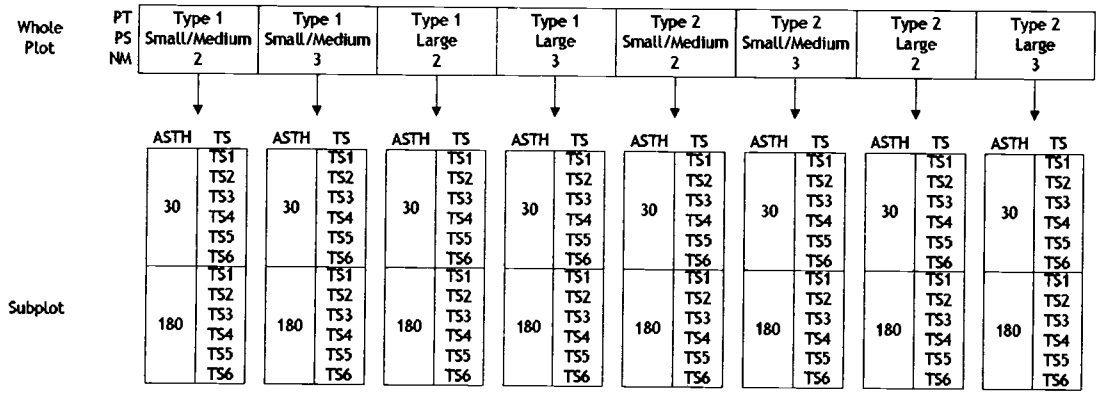

Figure 8.2. A Split–Plot Design with Five Design Factors

For each of the combinations of the whole plot factors, four problem data are generated randomly, each with a *different* number of board groups. For instance, when experimenting with small/medium size problem instances, four problem data with, for example, 3, 5, 6, and 8 board groups are generated. The same applies to large size problem instances as well. Consequently, $4 \times 4 = 16$ small/medium and 16 large size instances constitute a total of 32 problem data generated for all of the $2^3 = 8$ combinations of the whole plot factors. All $2 \times 6 = 12$

combinations of ASTH and TS heuristic factors are applied to the same problem data and the experiment is performed for minimizing the makespan and the mean (total) flow time, resulting in a total of $32 \times 12 \times 2 = 768$ computer runs.

The mathematical programming formulations MILP1 and MILP3, as well as the proposed B&P algorithm are also applied to each problem data for the two levels of ASTH. MILP1 is included since the proposed B&P algorithm is based on it, while MILP3 is included since it is the fastest among the three MILP formulations. Since the behavior of MILP2 is anticipated to be similar to that of MILP1, MILP2 is omitted in this experiment to reduce the experimental effort. As described previously in section 4.4, MILP3 cannot be applied to large size instances because of limited computational memory. Therefore, for small/medium instances, 16 (problem data) $\times$ 2 (levels of ASTH) $\times$ 3 (MILP1, MILP3, B&P) = 96 additional computer runs are performed. Similarly, for large size instances, 16 (problem data) $\times$ 2 (levels of ASTH) $\times$ 2 (MILP1, B&P) = 64 additional computer runs are performed. As a result, there are a total of 320 computer runs (160 per measure of performance) involving the solution of MILP1, MILP3, and the B&P algorithm. To keep this experiment manageable, the computation time is limited to five hours, which is also appropriate for real industry settings, when solving the problems with MILP1, MILP3, or the B&P algorithm.

The conventional approach used in the literature is comparing different heuristic or exact algorithms with respect to solution values and/or computation times. An experimental design with several factors such as different problem types and problem sizes makes sense in identifying the algorithms that outperform the others under *all* circumstances. However, it does not make much sense to try to determine whether certain factors significantly affect the solution values or computation times. In Amini and Barr [7], for example, all the factors (problem type,

problem size, type of change, and percentage of change) and their interactions are found to be significant. But, it is intuitively clear even before the conduct of the experiment that the problem size factor, for instance, will affect the computation time significantly. Indeed, in Amini and Barr [7], all the factors and their interactions turn out to be statistically significant, and the authors compare the reoptimization algorithms for each combination of the factors in their design separately.

The response tested in this dissertation is the percentage deviation of the solution values identified by the TS heuristics from the proposed lower bounds. It is most likely that the percentage deviation will increase as the problem size increases. However, it is not immediately clear how different magnitudes of setup times on the machines would effect the percentage deviation. In addition, the primary objective here is to *quantify* the performance of the TS heuristics. The goal is to put a yardstick on the performance across all circumstances considered in this dissertation. By doing so, the practitioner in the industry setting can be confident that the quality of the solution identified by the TS heuristic is quite close to the actual optimal solutions, i.e., the heuristic solution is only $\epsilon\%$ off the actual optimal solution where ϵ is to be estimated by the analysis of the experimental results.

The statistical model used to relate the percentage deviation to the factors and sources of error encapsulated by the split–plot design is as follows on the next page. The indices used in the model are defined as follows.

i = levels of problem type factor, $i = 1, 2$,

j = levels of problem size factor, $j = 1, 2$,

k = levels of number of machines factor, $k = 1, 2$,

l = index for the problem data, $l = 1, \ldots, 4$,

p = levels of ASTH factor, $p = 1, 2$,

q = levels of tabu search heuristic factor, $q = 1, \ldots, 6$,

$$
\begin{aligned}
Y_{ijklpq} = \mu &+ PT_i + PS_j + NM_k \\
&+ (PT{\times}PS)_{ij} + (PT{\times}NM)_{ik} + (PS{\times}NM)_{jk} + (PT{\times}PS{\times}NM)_{ijk} \\
&+ D_{l(ijk)} + ASTH_p + TS_q + (ASTH{\times}TS)_{pq} \\
&+ (PT{\times}ASTH)_{ip} + (PS{\times}ASTH)_{jp} + (NM{\times}ASTH)_{kp} \\
&+ (PT{\times}TS)_{iq} + (PS{\times}TS)_{jq} + (NM{\times}TS)_{kq} \\
&+ (PT{\times}ASTH{\times}TS)_{ipq} + (PS{\times}ASTH{\times}TS)_{jpq} + (NM{\times}ASTH{\times}TS)_{ipq} \\
&+ (PT{\times}PS{\times}ASTH)_{ijp} + (PT{\times}NM{\times}ASTH)_{ikp} + (PS{\times}NM{\times}ASTH)_{jkp} \\
&+ (PT{\times}PS{\times}TS)_{ijq} + (PT{\times}NM{\times}TS)_{ikq} + (PS{\times}NM{\times}TS)_{jkq} \\
&+ (PT{\times}PS{\times}ASTH{\times}TS)_{ijq} + (PT{\times}NM{\times}ASTH{\times}TS)_{ikq} \\
&+ (PS{\times}NM{\times}ASTH{\times}TS)_{jkq} + (PT{\times}PS{\times}NM{\times}ASTH)_{ijkp} \\
&+ (PT{\times}PS{\times}NM{\times}TS)_{ijkl} + (PT{\times}PS{\times}NM{\times}ASTH{\times}TS)_{ijkpq} \\
&+ (ASTH{\times}D)_{pl(ijk)} + (TS{\times}D)_{ql(ijk)} + E_{pql(ijk)}
\end{aligned}
$$

where

Y_{ijklpq} = the percentage deviation from the lower bound, $i = 1, 2$; $j = 1, 2$; $k = 1, 2$; $l = 1, \ldots, 4$; $p = 1, 2$; $q = 1, 2$,

μ = the mean percentage deviation,

PT_i = the main effect of problem type, $i = 1, 2$,

PS_j = the main effect of problem size, $j = 1, 2$,

NM_k = the main effect of number of machines, $k = 1, 2$,

$D_{l(ijk)}$ = the effect of problem data per problem type i, problem size j, and number of machines k, $l = 1, \ldots, 6$; $i = 1, 2$; $j = 1, 2$; $k = 1, 2$,

$ASTH_p$ = the main effect of average setup time per feeder on HSPM, $p = 1, 2$,

TS_q = the main effect of tabu search heuristic, $q = 1, \ldots, 6$,

$E_{pql(ijk)}$ = error term, $i = 1, 2$; $j = 1, 2$; $k = 1, 2$; $l = 1, \ldots, 4$; $p = 1, 2$; $q = 1, \ldots, 6$.

The remaining terms in the model are two– to five–factor interactions. Since the problem data is generated randomly, $D_{l(ijk)}$ are random effects, while the remaining effects are fixed. Hence, this is a *mixed* model with both random and fixed effects. The term $D_{l(ijk)}$ constitutes the whole plot error; the main effects and interactions of the whole plot factors are tested against it. The interaction term $(D \times ASTH \times TS)_{pql(ijk)}$ is pooled with $E_{pql(ijk)}$ to form the subplot error.

The experiments are performed in the following order. (1) A combination of the whole plot factor levels (problem type, problem size, and number of machines) is randomly selected. (2) Four problem data are generated consistent with the combination of the whole plot factor levels chosen in step 1. (3) All twelve combinations of the levels of the subplot factors (ASTH and TS heuristic) are applied to each of the four problem data in a random order.

8.3. Computing Platform and Software

All the experiments are run on a Dell®PowerEdge 2650 server with 4 GB of memory and dual Intel®Xeon™ processors operating at 2.4 GHz.

The six tabu search algorithms are implemented with C programming language using the Microsoft®Visual Studio.NET Framework 1.1 version 1.1.4322 SP1.

The mathematical programming models are solved with the optimization software ILOG®CPLEX™ using ILOG Concert Technology™ [66]. The B&P algorithms are implemented with C++ programming language using ILOG MAESTRO™ 1.0 [67], which is a branch–and–price–and–cut and shortest path optimization library, along with Concert Technology.

The analyses of experiments are performed with STATGRAPHICS® PLUS [112] and The SAS®System for Windows [104] statistical packages.

8.4. Results and Discussion

8.4.1. Tabu Search versus Actual Optimal Solutions

One of the objectives of the computational experiments is to quantify the performance of the tabu search heuristics with respect to the actual optimal solutions. The attempts made to identify the actual optimal solutions with CPLEX using the proposed mathematical programming formulations were successful only for two–machine small/medium size problems of type 2. As shall be presented later, in all other combinations of the whole plot factors, there were instances that could not be solved optimally with CPLEX within the imposed time limit of five hours. Hence, the analysis here is restricted to two–machine small/medium size problems of type 2 and the statistical inferences apply to such problems only. Note that, for this combination of the whole plot factors, the experimental model given in the previous section reduces to a randomized complete block design with two factors, ASTH and TS, and the details of the corresponding statistical model and the methods of analysis can be found in Montgomery [95].

The optimal solutions are obtained by solving MILP3 with CPLEX since MILP3 could be solved faster than the other two MILP formulations proposed in

chapter 4. Twelve problem data are generated and solved, for each level of ASTH, by all of the six TS heuristics and MILP3. The details of the experimental results are presented in Tables D.1 through D.5 in Appendix D.1. Consistent with the guidelines suggested by the aforementioned papers, the best/optimal solutions values and execution times, as well as the initial solution values and the times the TS heuristics and CPLEX *first* identify the best solutions they report are recorded.

Table 8.2 exhibits the average percentage deviations of the TS heuristic solutions from the optimal solutions for the two levels of ASTH. Each row represents the average percentage deviation for three problems. An entry with a $(\cdot)$ sign indicates that the TS heuristic identified the actual optimal solution to all three problems. Observing the detailed results presented in Tables D.2 through D.5 in Appendix D, it can be concluded that the TS heuristics performed extremely well. For minimizing the total flow time at least one of the TS heuristics identified the actual optimal solution for all 24 problems. Similarly, for minimizing the makespan, at least one of the proposed TS heuristics identified the actual optimal solution for 21 out of 24 problems and the solutions identified for the remaining three instances by the TS heuristics are quite close to the actual optimal solutions.

8.4.1.1. Minimizing the Total Flow Time

The analysis of variance (ANOVA) method is used to determine whether the effects of the two factors on the percentage deviation of the TS solutions from the actual optimal solutions are statistically significant or not. ANOVA requires certain assumptions to be satisfied. The normality of the percentage deviation assumption seems to be satisfied. However, the assumption of equal variances

Table 8.2. Average Percentage Deviations of TS Solutions from the Optimal Solutions

ASTH	N	Total Flow time						Makespan					
		TS1	TS2	TS3	TS4	TS5	TS6	TS1	TS2	TS3	TS4	TS5	TS6
180	3	.†	·	·	·	·	·	·	·	·	·	·	·
	5	·	·	·	·	·	·	0.92	·	0.92	0.92	·	·
	6	0.22	·	0.22	·	·	·	1.40	·	0.66	·	·	·
	8	2.06	0.61	1.13	0.79	0.61	0.61	1.94	0.99	0.79	0.30	0.99	0.99
30	3	·	·	·	·	·	·	·	·	·	·	·	·
	5	·	·	·	·	·	·	1.51	·	0.79	·	·	·
	6	·	·	·	·	·	·	0.67	0.42	0.67	·	0.42	0.42
	8	0.71	·	0.36	0.36	·	·	0.35	0.26	0.35	0.25	0.26	0.26

† Each cell is the average over three problems. A (·) sign indicates that the TS heuristic identified the actual optimal solution to all three problems.

seems to be violated and suggests a log–transformation (base 10) of the percentage deviation. The response, which is the percentage deviation of the TS solutions from the actual optimal solutions, is denoted by PD. The transformed response is $\log_{10}(\text{PD}+1)$, which for simplicity is denoted by logPD. The normal probability plots and the plots of residuals versus the factor levels after the log–transformation are also presented in Appendix D.1.

The results of ANOVA are presented in Table 8.3 for minimizing the total flow time. The effect of ASTH to logPD is statistically significant. Although suggestive (p–value is 0.0583), the effect of TS heuristic is not significant at a significance level of 0.05 (the observations are consistent with the null hypothesis that the TS heuristic factor does not effect logPD). After all, majority of the problems for both levels of ASTH are solved optimally with the majority of the TS heuristics. Similarly, the interaction of ASTH and TS heuristic is not significant.

Table 8.3. ANOVA Results for TS versus Optimal (Total Flow Time)

Source	Sum of Squares	Degrees of Freedom	Mean Square	F–value	p–value
Block (Problem)	0.700	11	0.065		
ASTH	0.083	1	0.083	8.15	0.0051
TS	0.110	5	0.022	2.20	0.0583
ASTH × TS	0.014	5	0.003	0.27	0.9287
Residual	1.24	121	0.010		
Corrected Total	2.15	143			

For minimizing the total flow time, 95% confidence intervals for the median[2] PD and mean logPD when ASTH=30 and ASTH=180 are presented in Table 8.4. When ASTH=30 seconds, it can be concluded with 95% confidence that the TS solutions are no different than the actual optimal solutions for the two–machine small/medium problems of type 2, since the percentage deviation of 0% is captured in the confidence interval. When ASTH=180 seconds, the median percentage deviation of the TS heuristic solutions from the actual optimal is only 0.164%. It is observed that the median PD increases slightly when the average setup time per feeder is increased from 30 seconds to 180 seconds. A 95% confidence interval, reported by Tukey's studentized range test [95], for the difference of the mean logPD when ASTH=30 and the mean logPD when ASTH=180 is [0.015, 0.081].

---

[2]If a log–transformation is applied to a response variable x and $\hat{\mu}(log_{10}(x))$ is an estimate of the mean of $log_{10}(x)$, then $10^{\hat{\mu}(log_{10}(x))}$ is interpreted as an estimate of the *median* of x.

Table 8.4. Confidence Intervals for the Two Levels of ASTH (Total Flow Time)

Levels of ASTH	Median PD	95% CI for Median PD	Mean logPD	95% CI for Mean logPD
180	0.164	[0.101, 0.230]	0.066	[0.042, 0.090]
30	0.042	[-0.013, 0.099]	0.018	[-0.006, 0.041]

Since the TS heuristics are not statistically different, it is concluded that using fixed/variable tabu list sizes or long term memory components based on LTM–max/LTM–min does not make a statistically significant difference in the percentage deviation of the solution values identified by the TS heuristics from the actual optimal solution values. However, incorporating variable tabu list sizes and/or long term memory components has a direct effect on the execution time of the TS heuristics. For example, the total time for a TS heuristic that uses a long term memory component with two more restarts is obviously more than that of a TS heuristic that does not use any long term memory components (no restarts). Table 8.5 presents the average execution time (TT) of the TS heuristics as well as the average time required by each of the TS heuristics to *first* identify the solution they eventually reported as the best one they found (TTB). Both figures are in seconds. The simplest TS heuristic is TS1 and uses fixed tabu list size and no long term memory components. Consequently, the TTB and TT values for TS1 are the smallest. TS2, on the other hand, uses variable tabu list size and no long term memory components. Observe that its total execution time, TT, is greater

than that of TS1. This is because the number of iterations without improvement[3] is reset to zero each time the tabu list size is increased or decreased, in order to give the heuristic more chance to identify better solutions with the modified tabu list size. As a result TS2 does not stop as quickly as TS1. Similarly, TS5 and TS6 use variable tabu list sizes but with long term memory components. Although TS3, TS4, TS5, and TS6 all use long term memory components, TS5 and TS6 use variable tabu list sizes (as opposed to fixed tabu list sizes used by TS3 and TS4). Consequently their total execution times are greater than that of TS3 or TS4.

Table 8.5. Average Execution Times of the TS Heuristics (Total Flow Time)

TS	Average TTB†	Average TT‡
TS1	0.012	0.023
TS3	0.023	0.079
TS4	0.061	0.209
TS2	0.075	0.535
TS5	0.075	0.558
TS6	0.090	0.569

† Time to best (TTB), in seconds.
‡ Total time (TT), in seconds.

In summary, since the ANOVA did not reveal any differences between the TS heuristics, it cannot be concluded using fixed or variable tabu list sizes makes a difference or not. Similarly, it cannot be concluded using long term memory components with LTM–max or LTM–min improve the percentage deviation or not. Although, all six TS heuristics are statistically the same with respect to the

---

[3]The search stops when the number of iterations without improvement reaches a certain predefined value.

percentage deviations of their solution values from the actual optimal solution values, TS2 seems to be the best choice among the six TS heuristics since it has the minimal median percentage deviation and its execution time is smaller than all the others (except for TS1 which has a highest median percentage deviation). TS1 claims superiority over all the other TS heuristics in terms of TTB and TT values. However, it has the worst median percentage deviation.

It is worth comparing the times required of the TS heuristics and the times required of CPLEX to solve MILP3. Since CPLEX uses branch–and–bound technique to solve the problems, even if it identifies the optimal solution, it spends most of its time for trying and proving the optimality of that solution. Therefore, it would not make much sense to compare the total execution time of CPLEX with the total execution times of the TS heuristics since the TS heuristics do not spend time to prove the optimality of the solutions they identify. However, from a practical point of view, if the question is to use CPLEX or the proposed TS heuristics to identify a good solution reasonably quickly, then it would make sense to compare TTB values for the TS heuristics and for CPLEX. In Table D.1 in Appendix D.1, the TTB values of CPLEX increase exponentially with the number of board groups in the problem and are much higher than the TTB values of any of the TS heuristics. For two–machine small/medium problems of type 2, the solutions identified by the TS heuristics are quite close to the actual optimal solutions and are identified extremely quickly.

8.4.1.2. Minimizing the Makespan

For minimizing the makespan, the data required a log–transformation, as well. Table 8.6 presents the results of ANOVA. The effect of TS heuristic is

statistically significant, while at a significance level of 0.05, the effect of ASTH is not, although it is suggestive (p–value is 0.083). The interaction of the two factors is not significant either.

Table 8.6. ANOVA Results for TS versus Optimal (Makespan)

Source	Sum of Squares	Degrees of Freedom	Mean Square	F–value	p–value
Block (Problem)	1.004	11	0.091		
ASTH	0.052	1	0.052	3.05	0.0833
TS	0.444	5	0.089	5.16	0.0003
ASTH × TS	0.039	5	0.008	0.45	0.8092
Residual	2.084	121	0.017		
Corrected Total	3.624	143			

As the levels of the TS heuristic are found to be significantly different, a second attempt is made to identify which of the TS heuristics are different than others. Table 8.7 presents the median PD and mean logPD values and associated 95% confidence intervals for each TS heuristic. As depicted in Figure 8.3, Tukey's studentized range test identified (based on logPD) that TS2–TS6 are not statistically different from each other. TS1 is statistically the same with TS3 but significantly different from the remaining TS heuristics as marked in Table 8.7. Note that the median percentage deviation for the TS heuristics are quite small (even the upper limits of the confidence intervals are all less than 0.8%), indicating that the TS heuristics solutions are highly close to the actual optimal solutions.

Since TS1, which is the simplest TS heuristic, only incorporates a fixed tabu list size and TS3 incorporates a fixed tabu list size with LTM–max, there is not an evidence that incorporating LTM–max makes a difference for fixed tabu

Table 8.7. Confidence Intervals for Different TS Heuristics and Homogenous Groups (Makespan)

TS	Avg. TTB†	Avg. TT†	Median PD	95% CI for Median PD	Homogenous Groups		Mean LogPD	95% CI for Mean logPD
TS4	0.038	0.089	0.138	[0.007, 0.285]	•		0.056	[0.003, 0.109]
TS2	0.027	0.061	0.156	[0.023, 0.306]	•		0.063	[0.010, 0.116]
TS5	0.027	0.169	0.156	[0.023, 0.306]	•		0.063	[0.010, 0.116]
TS6	0.026	0.170	0.156	[0.023, 0.306]	•		0.063	[0.010, 0.116]
TS3	0.008	0.061	0.417	[0.253, 0.600]	•	•‡	0.151	[0.098, 0.204]
TS1	0.007	0.022	0.577	[0.396, 0.782]		•	0.199	[0.145, 0.251]

† Average total time to best (TTB) and total time (TT), in seconds, over all 24 runs.
‡ TS heuristics with a • mark in the same column are statistically not different.

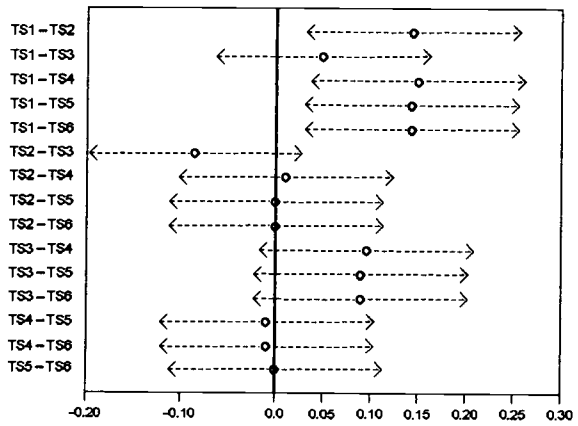

Figure 8.3. Simultaneous 95% Confidence Intervals Comparing Different TS Heuristics (Makespan)

list size heuristics. TS1 seems to be the worst among the six TS heuristics in terms of the percentage deviation of its solution values from the actual optimal solution values. TS3, although statistically not different from TS1, performed slightly better than TS1. However, since TS1 is statistically different from TS2,

TS4, TS5, and TS6 there is strong evidence that using the variable tabu list sizes only or long term memory components (especially LTM–min) with variable tabu list sizes improve the solutions identified by a TS heuristic.

Table 8.7 also presents the average execution time (TT) of the TS heuristics as well as the average time required by each of the TS heuristics to *first* identify the solution they eventually reported as the best one they found (TTB). Both figures are in seconds. Observe that TS4 has the minimal median percentage deviation, while it takes the longest for TS4 to identify the best solution compared to the other TS heuristics. This is primarily because TS4 incorporates long term memory with minimal frequency (LTM–min). This helps identify better solutions in the restarts after constructing new initial solutions, but obviously increases the time that the best solution was first identified (TTB) and the total execution time (TT). Also note that TS6, which incorporates variable tabu list sizes with LTM–min, has the longest TT. The reason for this is using a variable tabu list sizes increases the execution time since after reducing or increasing the tabu list size, the number of iterations without improvement is reset to 0 to give the heuristic more chance to identify better solutions with the modified tabu list size. Consequently, it takes longer for TS6 to stop the search.

In summary, for minimizing the makespan for the two–machine small/medium problems of type 2, TS2–TS6 are identified to be statistically the same in the percentage deviations of their solution values from the actual optimal solution values. However, TS4 seems to be the best choice since its solution values have the minimal median percentage deviation (see Table 8.7) and its execution time is quite reasonable compared to that of TS2, TS3, TS5, and TS6.

8.4.2. Lower Bounds versus Actual Optimal Solutions

Another objective of the computational experiments is to quantify the performance of the proposed lower bounds with respect to the actual optimal solutions. As in the previous section, since the actual optimal solutions can only be identified for small size problems in a timely manner, the analysis here is restricted to two-machine small/medium size problems of type 2 and the statistical inferences apply to such problems only. The results are presented in Tables D.6 and D.7 in Appendix D.2 in detail. Table 8.8 exhibits the average percentage deviations of the proposed lower bounds from the optimal solutions for the two levels of ASTH.

Table 8.8. Average Percentage Deviations of the Lower Bounds from the Optimal Solutions

ASTH	N	Total Flow Time	Makespan
180	3	0.00†	1.24
	5	1.32	0.99
	6	0.54	0.37
	8	1.31	1.07
30	3	1.84	0.27
	5	3.22	0.83
	6	2.24	0.35
	8	1.99	1.32

† Each cell is the average over three problems.

Each row in Table 8.8 presents the average percentage deviation for three problems. Observing these results, it can be concluded that the lower bounds are quite close to the actual optimal solution values.

8.4.2.1. Minimizing the Total Flow Time

The observations for the case of minimizing the total flow time satisfy the assumptions of ANOVA. The plots used for model adequacy checking are also presented in Appendix D.2. Table 8.9 below presents the results of ANOVA.

Table 8.9. ANOVA Results for Lower Bound versus Optimal (Total Flow Time)

Source	Sum of Squares	Degrees of Freedom	Mean Square	F–value	p–value
Block (Problem)	11.92	11	1.08		
ASTH	8.32	1	8.32	6.64	0.026
Residual	13.79	11	1.25		
Corrected Total	34.03	23			

The effect of ASTH is significant on the percentage deviation of the lower bounds from the actual optimal solution values. The mean percentage deviation of the lower bounds from the actual optimal solutions for ASTH=30 is 2.13% and the corresponding 95% confidence interval is [1.42%, 2.84%]. For ASTH=180, the mean percentage deviation is 0.95% and the corresponding 95% confidence interval is [0.24%, 1.66%]. Therefore, for two–machine small/medium size problems of type 2, the lower bounds are quite close to the actual optimal solution values.

8.4.2.2. Minimizing the Makespan

The observations for the case of minimizing makespan also satisfy the assumptions of ANOVA. The plots used for model adequacy checking are also presented in Appendix D. The results of ANOVA are presented in Table 8.10 below.

Table 8.10. ANOVA Results for Lower Bound versus Optimal (Makespan)

Source	Sum of Squares	Degrees of Freedom	Mean Square	F–value	p–value
Block (Problem)	10.77	11	0.98		
ASTH	0.30	1	0.30	0.47	0.505
Residual	6.99	11	0.64		
Corrected Total	18.08	23			

In the case of minimizing the makespan, the effect of ASTH on the percentage deviation of the lower bounds from the actual optimal solution values is not statistically significant. Over both levels of ASTH, the mean percentage deviation of the lower bounds from the actual optimal solutions is only 0.80% with the corresponding 95% confidence interval computed as [0.43%, 1.17%]. It is concluded that the lower bounds are extremely close to the actual optimal solutions for two–machine small/medium size problems of type 2.

8.4.2.3. Comparison with the Procedure Minsetup Based Lower Bounds

In this section, the lower bounds obtained from the B&P algorithm are compared with the lower bounds based on procedure Minsetup, again for two–machine small/medium size problems of type 2. Appendix D.2 also presents the lower bounds based on procedure Minsetup. Figure 8.4 presents the lower bounds from the two methods on the same graph for minimizing the total flow time with each of the two levels of ASTH. Similarly, Figure 8.5 compares the lower bounds from both methods for minimizing the makespan with each of the two levels of ASTH.

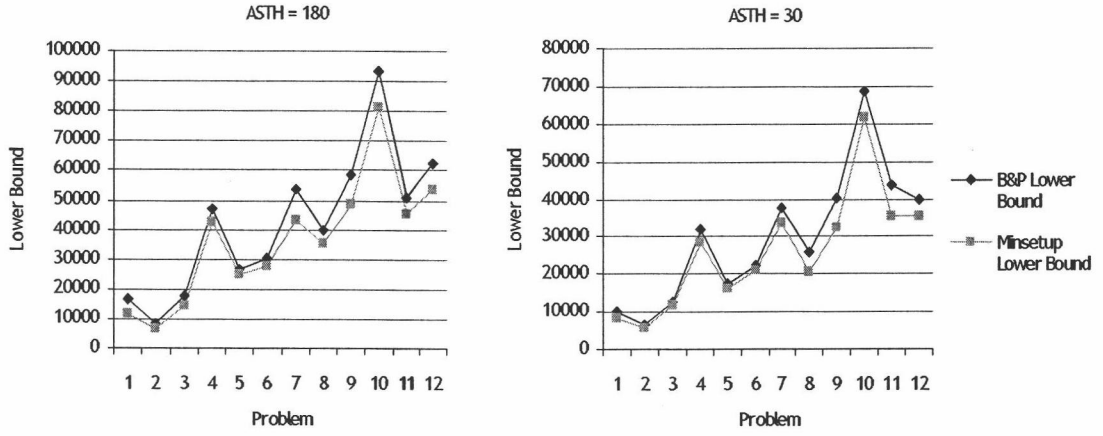

Figure 8.4. Comparison of B&P and Procedure Minsetup Based Lower Bounds (Total Flow Time)

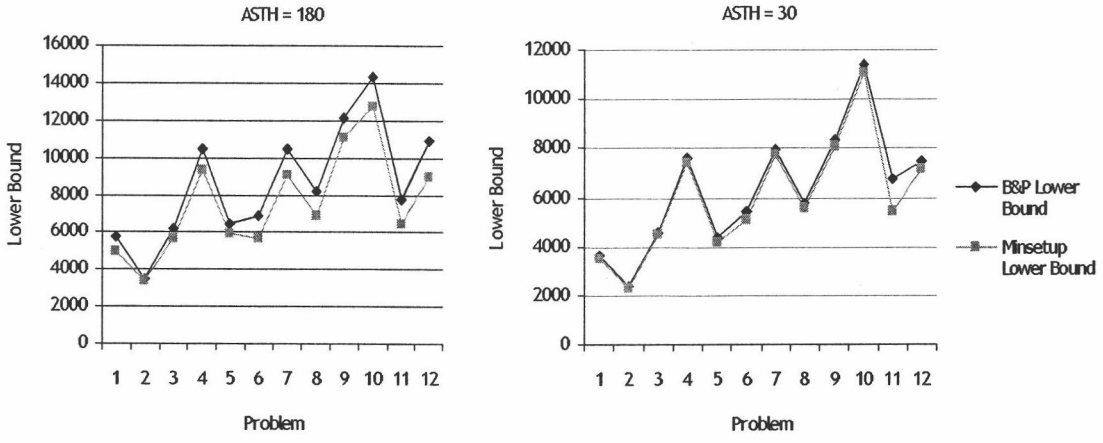

Figure 8.5. Comparison of B&P and Procedure Minsetup Based Lower Bounds (Makespan)

It is observed that the lower bounds based on procedure Minsetup are consistently inferior to the lower bounds identified by the B&P algorithm. Procedure Minsetup converts the carryover sequence–dependent setup times problem to one with sequence–independent setup times. As a result of this conversion, the actual setup times are poorly approximated and the resulting lower bounds are not as

high. B&P algorithm, on the other hand, makes use of the actual setup times with carryover.

In addition, the difference between the Minsetup based lower bounds and B&P based lower bounds is higher for minimizing the total (mean) flow time than for minimizing the makespan. This is most likely because of the fact that, for minimizing the total (mean) flow time, the value of the lower bound depends highly on the minimum run times and minimum setup times (refer to page 97 for the definitions of r_1 and r_2). As a result, as the number of board groups and board types increase, the lower bounds from procedure Minsetup degrade rapidly and the difference between the two types of lower bounds increase.

This behavior is anticipated to be similar for other types of problems tested in the experiments. Therefore, procedure Minsetup based lower bounds are not considered in the analyses that follow in the remainder of this chapter.

8.4.3. Tabu Search versus Lower Bounds

The most important objective of the computational experiments is to quantify the performance of the TS heuristics with respect to the proposed lower bounds. This section presents the results of the split–plot experimental design described previously. The detailed results including the SAS codes used to analyze the results are presented in Appendix D.3. In the sequel, rigorous statistical analyses are carried out in order to identify significant differences, if any, among the proposed TS heuristics in terms of the percentage deviations of the solution values from the proposed lower bounds and among the factors that affect the percentage deviation, and in order to estimate the percentage deviation of the TS solutions from the proposed lower bounds.

8.4.3.1. Minimizing the Total Flow Time

Table 8.11 summarizes the percentage deviations of the six TS solutions for minimizing the total flow time for each *combination* of the experimental factors. Table 8.12, on the other hand, summarizes the percentage deviations of the six TS solutions for minimizing the total flow time for each *level* of the experimental factors.

Table 8.11. Average Percentage Deviations of TS Solutions from the Lower Bounds for Each Combination of the Experimental Factors (Total Flow Time)

ASTH	NM†	Size‡	Type	TS1	TS2	TS3	TS4	TS5	TS6
180	2	S/M	1	1.40	1.06	1.22	1.06	1.06	1.06
			2	0.76	0.37	0.76	0.76	0.37	0.37
		L	1	9.99	8.47	8.98	8.47	8.47	8.47
			2	8.22	7.10	7.55	7.13	7.10	7.10
	3	S/M	1	4.60	3.87	4.17	3.87	3.87	3.87
			2	1.63	0.72	0.72	0.72	0.72	0.72
		L	1	11.97	10.28	11.09	10.30	10.28	10.28
			2	8.62	6.33	6.99	6.91	6.33	6.33
30	2	S/M	1	2.90	2.64	2.90	2.64	2.64	2.64
			2	2.19	2.19	2.19	2.19	2.19	2.19
		L	1	6.66	6.43	6.49	6.40	6.40	6.40
			2	5.19	3.22	4.31	3.25	3.22	3.22
	3	S/M	1	2.99	2.83	2.75	2.00	2.64	2.10
			2	1.91	1.73	1.80	1.73	1.73	1.73
		L	1	7.61	6.89	7.33	6.88	6.87	6.87
			2	5.07	3.95	4.00	3.95	3.95	3.95
Overall Average				5.15	4.25	4.58	4.27	4.24	4.20

† NM: The number of machines

‡ S/M: small/medium size, L: large size

Table 8.12. Average Percentage Deviations of TS Solutions for Each Level of the Experimental Factors (Total Flow Time)

Factor	Level	TS1	TS2	TS3	TS4	TS5	TS6	Overall
ASTH	180	5.90	4.78	5.18	4.90	4.78	4.78	5.05
	30	4.32	3.71	3.97	3.63	3.71	3.64	3.83
m†	2	4.67	3.94	4.30	3.99	3.93	3.93	4.13
	3	5.55	4.57	4.86	4.55	4.57	4.48	4.75
Size‡	S/M	2.30	1.90	2.06	1.87	1.90	1.84	1.98
	L	7.92	6.59	7.09	6.66	6.58	6.58	6.90
Type	1	6.02	5.30	5.62	5.20	5.29	5.21	5.43
	2	4.20	4.19	3.54	3.33	3.20	3.20	3.45

† m: The number of machines

‡ S/M: small/medium size, L: large size

The following observations are made regarding the summary of the results. Although the overall average percentage deviations of all six TS heuristics are close to each other, TS heuristics incorporating variable tabu list size (TS2, TS5, and TS6) and TS4, which uses fixed tabu list size and long term memory with diversification, seem to consistently dominate TS1 and TS3. The simplest TS heuristic, TS1, has the worst average percentage deviation as expected. There is a clear distinction between the percentage deviations of small/medium and large size problems. As anticipated, the percentage deviations increase as the size of the problems increase. The type of a problem also seems to have a profound effect on the percentage deviation; the average percentage deviation of the type 1 problems is higher than that of the type 2 problems. This is also anticipated since type 2 problems only have one or two individual board types per board group, while in type 1 problems there are three to five individual board types per board group. Therefore, the size of a type 1 problem is larger than that of a type 2 problem,

even if the number of board types are the same in both problems. The average percentage deviation of the TS solutions from the lower bounds is only slightly higher for three–machine problems than for the two–machine problems. The effect of the average setup time per feeder, ASTH, factor also seems to be significant. The percentage deviations increase as ASTH is increased from 30 seconds to 180 seconds. This may be because of the fact that the range of the completion times of the new columns to be generated during the column generation/B&P algorithm depends on the magnitudes of the setup and run times on the machines, and the number of columns need to be generated increase as the range of the completion times increase. Most importantly, though, the LP–relaxation of MILP1 is much tighter when ASTH is 30 seconds than it is 180 seconds for both minimizing the total flow time and the makespan. Since the proposed lower bounds are as good as the LP–relaxation values of MILP1, they are much better when ASTH is 30 seconds (refer to Figure 8.8 on page 203 for a detailed explanation on this subject).

Rigorous statistical analyses are performed to test the validity of these observations. Since all the assumptions of ANOVA seem to be satisfied (refer to Appendix D.3 for model adequacy checking), the analysis is carried out with ANOVA to determine which of the experimental factors are statistically significant and which are not. Table 8.13 presents the results of ANOVA.

As expected, the main effect of the problem type (PT) and problem size (PT) factors are found to be significant. The main effect of the number of machines factor (NM) and all the other interactions of the whole plot factors are not statistically significant. In the subplot level, the main effects of the subplot factors ASTH and TS are both significant. Also, the interaction terms $PS \times ASTH$ $PS \times TS$, and $PT \times PS \times TS$ are found to be significant.

Table 8.13. ANOVA Results for Tabu Search versus Lower Bounds (Total Flow Time)

Source	Sum of Squares	Degrees of Freedom	Mean Square	F–value	p–value
Whole Plot					
PT	388.111	1	388.111	10.12	0.0040
PS	2332.630	1	2332.630	60.85	<0.0001
NM	37.419	1	37.419	0.98	0.3330
$PT{\times}PS$	46.977	1	46.977	1.23	0.2793
$PT{\times}NM$	44.356	1	44.356	1.16	0.2928
$PS{\times}NM$	0.138	1	0.138	0.00	0.9527
$PT{\times}PS{\times}NM$	0.564	1	0.564	0.01	0.9045
$D(PT{\times}PS{\times}NM)$	920.035	24	38.335		
Subplot					
$ASTH$	138.132	1	138.132	4.58	0.0427
TS	45.224	5	9.045	53.70	<0.0001
$ASTH{\times}TS$	1.797	5	0.359	2.24	0.0547
$PS{\times}ASTH$	336.170	1	336.170	11.15	0.0027
$PT{\times}ASTH$	5.595	1	5.595	0.19	0.6705
$NM{\times}ASTH$	26.026	1	26.026	0.86	0.3622
$PT{\times}TS$	0.535	5	0.107	0.64	0.6729
$PS{\times}TS$	12.538	5	2.508	14.84	<0.0001
$NM{\times}TS$	1.313	5	0.263	1.56	0.1771
$PT{\times}PS{\times}ASTH$	14.057	1	14.057	0.47	0.5013
$PT{\times}NM{\times}ASTH$	32.977	1	32.977	1.09	0.3061
$PS{\times}NM{\times}ASTH$	16.356	1	16.356	0.54	0.4686
$PT{\times}PS{\times}TS$	2.627	5	0.525	3.12	0.0111
$PT{\times}NM{\times}TS$	0.983	5	0.197	1.17	0.3296
$PS{\times}NM{\times}TS$	0.770	5	0.154	0.91	0.4746
$PT{\times}ASTH{\times}TS$	0.465	5	0.093	0.58	0.7151
$PS{\times}ASTH{\times}TS$	0.324	5	0.065	0.40	0.8456
$NM{\times}ASTH{\times}TS$	1.117	5	0.223	1.39	0.2321
$PT{\times}PS{\times}NM{\times}ASTH$	0.032	1	0.032	0.00	0.9744
$PT{\times}PS{\times}NM{\times}TS$	0.983	5	0.197	1.17	0.3292
$PT{\times}PS{\times}ASTH{\times}TS$	1.141	5	0.228	1.42	0.2212
$PT{\times}NM{\times}ASTH{\times}TS$	0.738	5	0.148	0.92	0.4704
$PS{\times}NM{\times}ASTH{\times}TS$	1.152	5	0.230	1.44	0.2163
$PT{\times}PS{\times}NM{\times}ASTH{\times}TS$	1.558	5	0.312	1.94	0.0924
$ASTH{\times}D(PT{\times}PS{\times}NM)$	723.785	24	30.158		
$TS{\times}D(PT{\times}PS{\times}NM)$	20.213	120	0.168		
$ASTH{\times}TS{\times}D(PT{\times}PS{\times}NM)$	19.26	120	0.160		

When comparing more than two means, ANOVA indicates whether the means are significantly different from each other, but does not indicate which means actually differ. The significance identified by ANOVA output results in the need to further conduct analysis to determine which pairs or groups of TS heuristics are statistically different in terms of the percentage deviations of their solutions from the lower bounds. Over all the levels of the experimental factors, Table 8.14 presents the estimates of the percentage deviations of each TS heuristic with the corresponding 95% confidence intervals and groups of TS heuristics as identified by Tukey's studentized range test [95]. Any two heuristics in the same homogenous group cannot be distinguished from each other. As observed, TS1 and TS3 are statistically the same while TS2–TS6 also cannot be distinguished. Note that, although statistically not different from TS2–TS5, the average percentage deviation of the TS6 solutions is numerically minimal.

Table 8.14. Average Percentage Deviations of TS Solutions and Homogenous Groups (Total Flow Time)

TS Heuristic	Mean PD	95% CI for Mean PD	Homogenous Groups	
TS6	4.20	[3.54, 4.86]	•	
TS5	4.24	[3.59, 4.90]	•	
TS2	4.25	[3.59, 4.91]	•	
TS4	4.26	[3.60, 4.92]	•	
TS3	4.58	[3.91, 5.22]	•	•†
TS1	5.15	[4.49, 5.80]		•

† TS heuristics with a • mark in the same column are statistically not different.

Observe in the ANOVA table that two interaction terms, $PS \times TS$ and $PT \times PS \times TS$, are found to be significant. Thus, the highest order significant

interaction term that includes the TS factor is $PT \times PS \times TS$. Since the most important experimental factor of interest is the TS factor, further analysis is performed to identify any differences among the six TS heuristics over the effect slices of this interaction term. As presented in Table 8.15, the TS heuristics are significantly different when large size problems with both types, and small/medium size problems of type 1 are considered.

Table 8.15. Tests of Slice Effects for Differences Among the Levels of TS (Total Flow Time)

Effect	PS†	PT†	Ndf ‡	Ddf ‡	F–value	p–value
$PT \times PS \times TS$	Large	Type 1	5	120	21.85	<0.0001
	Small/Med.	Type 1	5	120	7.18	<0.0001
	Large	Type 2	5	120	42.32	<0.0001
	Small/Med.	Type 2	5	120	0.99	0.4266

† *PT*: Problem Type, *PS*: Problem Size (large, small/medium)
‡ Ndf: Numerator degrees of freedom, Ddf: Denominator degrees of freedom

Since significant differences are found, *multiple comparisons*, using Tukey's studentized range test, are performed for the combinations of the problem size (PS) and problem type (PT) factors. Table 8.16 presents the estimates of the percentage deviations of the TS heuristics for all four combinations of these factors. The TS heuristics with a '•' mark in the same row are statistically not different. For example, for large size problems of type 1, TS2–TS6 cannot be statistically distinguished. TS1 in this case is statistically inferior than the remaining five TS heuristics. In general, the numerical value of the percentage deviation is smaller when variable tabu list sizes and long term memory components are used (TS2–TS6).

Table 8.16. Percentage Deviations and Homogenous Groups of TS Heuristics for Different Problem Sizes and Types (Total Flow Time)

PS	PT	TS1	TS2	TS3	TS4	TS5	TS6
Large	Type 1	• 9.18	• 8.00	• 8.44	• 7.99	• 7.99	• 7.99
Small/Med.	Type 1	• 3.13	• 2.55	• • 2.76	• 2.39	• 2.55	• 2.41
Large	Type 2	• 6.86	• 5.20	• 5.70	• • 5.30	• 5.14	• 5.14
Small/Med.	Type 2	• 1.51	• 1.25	• 1.37	• 1.35	• 1.25	• 1.25

† TS heuristics with a • mark in the same row are statistically not different.

In summary, TS1, which is the simplest TS heuristic using a fixed tabu list size and no long–term memory component, and TS3, which uses a fixed tabu list size and long–term memory based on LTM–max, are consistently dominated by the other TS heuristics. That is, there is strong evidence that using variable tabu list size and long–term memory based on LTM–min to diversify the search have a strong impact on the quality of the solutions obtained by the TS heuristics. The differences between these two groups of TS heuristics are pronounced more for large size (and type 1) problems. This is expected since the solution space of the problem grows exponentially with the size of the problem in terms of the number of board groups and individual board types within groups. Variable tabu list sizes provide a more powerful means to scan the solution space for better solutions, as TS2 is found to be significantly better than TS1. The only difference between the two heuristics is that TS1 uses fixed tabu list size and TS2 uses variable tabu list

size, and both heuristics do not employ any long term memory component. The variable tabu list size provides more flexibility for searching the solution space, while using long term memory based on LTM–min easily shifts the search to the regions of the solution space that have not been visited before. Therefore, the chances of finding better solutions increase, especially when variable tabu list size and long term memory based on LTM–min are employed simultaneously as in TS6 (TS6 has the minimal percentage deviation value in almost all cases). Using long term memory based on LTM–max only deepens the search within the regions visited previously, and as the solution space enlarges, it is likely that better solutions can be found at other regions of the solution space.

Naturally, incorporation of variable tabu list sizes and/or long term memory components requires additional computational effort. Therefore, the simplest TS heuristic, TS1, is computationally the least demanding. TS3, for instance, incorporates long term memory based on LTM–max and performs two more restarts to further search the solution space. Therefore, TS3 requires more computation time compared to TS1. However, each restart may –and most likely– take a different amount of time to complete because of the stopping criteria employed. For instance, the search stops when the best solution cannot be improved for a certain number of iterations. It is likely that the later restarts take shorter amounts of time since the best solution found so far keeps improving, reducing the chances of finding even better solutions later in the search process. In sum, the computational effort demanded by each TS heuristic differs. Table 8.17 presents the average computation time of each TS heuristic for each combination of the other experimental factors. As expected, the computation times are higher for large size problems. As the number of groups increase, so does the number of possible sequences, which are generated, evaluated, and compared by the heuristics. There is

a clear difference between the computation times for type 1 and type 2 problems. Since there are more individual board types within groups in type 1 problems, inside tabu search (that tries to find better sequences of individual board types within the groups) takes longer for type 1 problems. In general, the computation time of the heuristics are a little bit higher for three–machine problems than for two–machine problems. For the same number of board groups and board types within groups, the number of possible sequences are the same implying that the heuristics have similar power to search the solution space of both problems. However, each sequence has to be evaluated over three machines rather than two, which increases the computation time.

Observe in Table 8.17 that the computation times required of TS2, TS5, and TS6 are higher than that of the other three heuristics. TS2, TS5, and TS6 all use variable tabu list sizes. Using variable tabu list sizes increases the computation time since after reducing or increasing the tabu list size, the number of iterations without improvement is reset to 0 to give the heuristic more chance to identify better solutions *with the modified tabu list size.* Consequently, it takes longer for TS2, TS5, and TS6 to stop the search. Additionally, in contrast with TS2, heuristics TS5 and TS6 restart the process two more times; hence the difference in the computation times.

Since, in general, there is not a statistically significant difference in the percentage deviations of TS2–TS6, TS3 seems to be the best choice since it has the minimal computation time. However, TS3 was significantly inferior than TS2, TS4, TS5, and TS6 for certain problem structures such as large size problems of type 2 (refer to Figure 8.16). As observed in the analysis of the experiments, TS2, TS4, TS5, and TS6 cannot be statistically distinguished for any of the problem structures. Therefore, it is concluded that TS4, with a smaller average

Table 8.17. Average Computation Times (in seconds) of TS Solutions (Total Flow Time)

ASTH	NM†	Size‡	Type	TS1	TS2	TS3	TS4	TS5	TS6
180	2	S/M	1	0.14	5.62	1.71	2.73	7.02	6.98
			2	0.02	0.55	0.14	0.26	0.84	0.88
		L	1	0.66	20.47	8.86	21.75	40.31	41.79
			2	0.23	5.16	0.93	2.04	6.43	6.42
	3	S/M	1	0.14	6.61	3.43	5.08	8.33	8.80
			2	0.06	0.45	0.16	0.23	0.89	0.95
		L	1	0.50	18.11	7.61	22.73	31.33	32.02
			2	0.28	5.60	1.02	3.34	7.16	7.17
30	2	S/M	1	0.10	4.79	1.06	2.06	6.83	7.20
			2	0.04	0.45	0.15	0.25	0.76	0.80
		L	1	0.58	23.02	3.47	36.90	54.24	55.04
			2	0.13	4.71	0.98	2.41	6.44	6.52
	3	S/M	1	0.25	11.81	1.85	5.12	20.55	22.13
			2	0.06	0.46	0.21	0.18	0.84	0.85
		L	1	0.86	20.19	7.02	19.16	40.94	42.28
			2	0.29	5.55	1.70	2.75	7.99	8.26
Overall Average				0.27	8.35	2.52	7.94	15.06	15.51

† NM: The number of machines

‡ S/M: small/medium size, L: large size

computation time, is the TS heuristic that should be employed for the kinds of scheduling problems addressed here.

The other most important experimental factor of interest is the ASTH factor, which significantly affects the percentage deviation of the TS heuristics. Over all problem structures, the percentage deviations are estimated to be 3.83% and 5.05% when ASTH is 30 and 180 seconds, respectively. In general, the percent-

age deviation of the TS heuristics seem to increase as ASTH is changed from 30 seconds to 180 seconds.

Observe in the ANOVA table that the only significant interaction term that includes ASTH is the $PS \times ASTH$. Therefore, additional insight is provided by the tests performed to identify any differences between the levels of the ASTH factor on the percentage deviation of the TS heuristics over the effect slices of this interaction. Table 8.18 presents the estimates of the percentage deviations of the TS heuristics for the two levels of ASTH separately for each slice (for each problem size).

Table 8.18. Tests of Slice Effects for Differences Among the Levels of ASTH (Total Flow Time)

PS	ASTH	
	30	180
Large	• 5.37	• 8.45
Small/Medium	• 2.32	• 1.68

The numerical value of the percentage deviation is smaller when ASTH=30 seconds. As presented in the same table, the differences are significant only when large size problems are considered. This may be because of the fact that the range of the completion times of the new columns to be generated by the column generation/B&P algorithm depends on the magnitudes of the setup and run times on the machines, and the number of columns need to be generated increase as the range of the completion times increase. Most importantly, though, the LP–relaxation of MILP1 is much tighter when ASTH is 30 seconds than it is 180

seconds. *For the same problem data*, the *gap* between the optimal solution and the LP–relaxation lower bound is generally much higher for ASTH=180 seconds than it is for ASTH=30 seconds. In other words, the LP–relaxation value of MILP1 is much closer to the optimal solution (hence closer to an an upper bound) when ASTH=30 seconds. This issue is discussed in the next section in detail (refer to Figure 8.8 on page 203). Since the proposed lower bounds are as good as the LP–relaxation values of MILP1, they are much better when ASTH is 30 seconds.

8.4.3.2. Minimizing the Makespan

Table 8.19 summarizes the percentage deviations of the six TS solutions for minimizing the makespan for each *combination* of the experimental factors. Table 8.20, on the other hand, summarize the percentage deviations of the six TS solutions for minimizing the total flow time for each *level* of the experimental factors.

As in the case of minimizing the total flow time, although the overall average percentage deviations of all six TS heuristics are close to each other, TS heuristics incorporating variable tabu list size (TS2, TS4, and TS5) and/or long term memory based on LTM–min (TS4 and TS6), seem to consistently dominate TS1 and TS3. As anticipated, the percentage deviations seem to increase as the size of the problems increase. The type of a problem also seems to have a profound effect on the percentage deviation; the average percentage deviation of the type 1 problems is higher than that of the type 2 problems (type 2 problems only have one or two individual board types per board group, while in type 1 problems there are three to five individual board types per board group). Again, the average percentage deviation of the TS solutions from the lower bounds is only slightly

Table 8.19. Average Percentage Deviations of TS Solutions from the Lower Bounds for Each Combination of the Experimental Factors (Makespan)

ASTH	$m^{\dagger}$	Size‡	Type	TS1	TS2	TS3	TS4	TS5	TS6
180	2	S/M	1	2.27	1.15	1.78	1.04	1.15	1.15
			2	2.32	1.10	1.77	1.50	1.10	1.10
		L	1	10.26	8.15	8.56	8.41	8.15	8.15
			2	12.51	8.14	11.54	9.08	8.14	8.14
	3	S/M	1	4.12	3.26	2.96	2.85	3.26	2.60
			2	3.02	1.96	2.92	1.96	1.96	1.96
		L	1	12.14	11.15	11.62	11.05	11.15	11.04
			2	9.23	7.29	8.06	7.29	7.29	7.29
30	2	S/M	1	1.29	0.98	1.29	1.04	0.98	0.98
			2	1.04	0.78	1.04	0.59	0.78	0.78
		L	1	7.28	7.10	7.20	7.18	7.10	7.10
			2	5.99	5.00	5.51	5.00	5.00	5.00
	3	S/M	1	3.82	2.61	2.99	2.50	2.61	2.61
			2	1.66	1.61	1.66	1.66	1.61	1.61
		L	1	9.04	8.89	8.92	8.94	8.89	8.86
			2	5.22	5.07	5.12	5.02	5.07	5.07
Overall Average				5.69	4.64	5.18	4.69	4.64	4.59

† m: The number of machines

‡ S/M: small/medium size, L: large size

higher for three–machine problems than for the two–machine problems. The effect of the average setup time per feeder, ASTH, factor also seems to be significant; the percentage deviations increase as ASTH is increased from 30 seconds to 180 seconds.

Rigorous statistical analyses are performed to identify factors that significantly affect the percentage deviations. Since all the assumptions of ANOVA

Table 8.20. Average Percentage Deviations of TS Solutions for Each Level of the Experimental Factors (Makespan)

Factor	Level	TS1	TS2	TS3	TS4	TS5	TS6	Overall
ASTH	180	6.98	5.27	6.15	5.40	5.27	5.18	5.71
	30	4.42	4.00	4.22	3.99	4.00	4.00	4.11
m†	2	5.37	4.05	4.84	4.23	4.05	4.05	4.43
	3	6.03	5.23	5.53	5.16	5.23	5.13	5.38
Size‡	S/M	2.44	1.68	2.05	1.64	1.68	1.60	1.85
	L	8.96	7.60	8.32	7.75	7.60	7.58	7.97
Type	1	6.28	5.41	5.66	5.38	5.41	5.31	5.57
	2	5.12	3.87	4.70	4.01	3.87	3.87	4.24

† m: The number of machines

‡ S/M: small/medium size, L: large size

seem to be satisfied (refer to Appendix D.3 for model adequacy checking), the analysis is carried out with ANOVA and the results are presented in Table 8.21 on the next page.

As expected, the main effect of the problem size factor (PS) is found to be significant. Surprisingly, although suggestive (p–value is 0.0609), the main effect of the problem type factor (PT) is not significant at a significance level of 0.05. The main effect of the number of machines factor (NM) and all the other interactions of the whole plot factors are not significant, either. The main effects of the subplot factors ASTH and TS, and the interaction terms $ASTH \times TS$, $PS \times ASTH$, $PT \times PS \times TS$, $PT \times ASTH \times TS$, and $PS \times ASTH \times TS$ are also found significant.

Table 8.21. ANOVA Results for Tabu Search versus Lower Bounds (Makespan)

Source	Sum of Squares	Degrees of Freedom	Mean Square	F–value	p–value
Whole Plot					
PT	170.187	1	170.187	3.87	0.0609
PS	3583.515	1	3583.515	81.43	<0.0001
NM	86.431	1	86.431	1.96	0.1739
$PT{\times}PS$	54.964	1	54.964	1.25	0.2748
$PT{\times}NM$	106.134	1	106.134	2.41	0.1335
$PS{\times}NM$	10.501	1	10.501	0.24	0.6296
$PT{\times}PS{\times}NM$	31.510	1	31.510	0.72	0.4058
$D(PT{\times}PS{\times}NM)$	1056.203	24	44.008		
Subplot					
$ASTH$	246.048	1	246.048	10.97	0.0029
TS	62.945	5	12.589	30.79	<0.0001
$ASTH{\times}TS$	23.431	5	4.686	12.85	<0.0001
$PS{\times}ASTH$	118.481	1	118.481	5.28	0.0305
$PT{\times}ASTH$	21.113	1	21.113	0.94	0.3415
$NM{\times}ASTH$	0.831	1	0.831	0.04	0.8490
$PT{\times}TS$	4.288	5	0.858	2.10	0.0704
$PS{\times}TS$	4.277	5	0.855	2.09	0.0710
$NM{\times}TS$	4.458	5	0.892	2.18	0.0607
$PT{\times}PS{\times}ASTH$	8.857	1	8.857	0.40	0.5356
$PT{\times}NM{\times}ASTH$	10.534	1	10.534	0.47	0.4996
$PS{\times}NM{\times}ASTH$	0.838	1	0.838	0.04	0.8483
$PT{\times}PS{\times}TS$	6.041	5	1.208	2.96	0.0149
$PT{\times}NM{\times}TS$	2.740	5	0.548	1.34	0.2520
$PS{\times}NM{\times}TS$	5.006	5	1.001	2.45	0.0376
$PT{\times}ASTH{\times}TS$	4.767	5	0.953	2.61	0.0279
$PS{\times}ASTH{\times}TS$	6.030	5	1.206	3.31	0.0078
$NM{\times}ASTH{\times}TS$	2.969	5	0.594	1.63	0.1576
$PT{\times}PS{\times}NM{\times}ASTH$	10.976	1	10.976	0.49	0.4909
$PT{\times}PS{\times}NM{\times}TS$	2.836	5	0.567	1.39	0.2338
$PT{\times}PS{\times}ASTH{\times}TS$	0.517	5	0.103	0.28	0.9212
$PT{\times}NM{\times}ASTH{\times}TS$	0.679	5	0.136	0.37	0.8666
$PS{\times}NM{\times}ASTH{\times}TS$	1.003	5	0.201	0.55	0.7380
$PT{\times}PS{\times}NM{\times}ASTH{\times}TS$	2.858	5	0.572	1.57	0.1745
$ASTH{\times}D(PT{\times}PS{\times}NM)$	538.102	24	22.421		
$TS{\times}D(PT{\times}PS{\times}NM)$	49.064	120	0.409		
$ASTH{\times}TS{\times}D(PT{\times}PS{\times}NM)$	43.757	120	0.365		

The significance identified by the ANOVA output results in the need to further conduct analyses to determine which pairs or groups of TS heuristics are statistically different in the percentage deviations of their solutions from the lower bounds. Over all the levels of the experimental factors, Table 8.22 presents the estimates of the percentage deviations of each TS heuristic with the corresponding 95% confidence intervals and groups of TS heuristics as identified by Tukey's studentized range test [95]. Any two heuristics in the same homogenous group cannot be distinguished from each other. As observed, TS1 is statistically different from all the other TS heuristics. Similarly, TS3 is significantly different from the other TS heuristics, while TS2, TS4, TS5, and TS6 form another homogenous group. Note that, the average percentage deviation of the TS6 solutions is numerically minimal, although the difference is not statistically different from either of TS2, TS4, or TS5.

Table 8.22. Average Percentage Deviations of TS Solutions and Homogenous Groups (Makespan)

TS Heuristic	Mean PD	95% CI for Mean PD	Homogenous Groups		
TS6	4.58	[3.87, 5.30]	•		
TS5	4.63	[3.92, 5.34]	•		
TS2	4.63	[3.92, 5.34]	•		
TS4	4.68	[3.97, 5.40]	•		
TS3	5.17	[4.46, 5.88]		•†	
TS1	5.69	[4.98, 6.40]			•

† TS heuristics with a • mark in the same column are statistically not different.

Observe in the ANOVA table that the highest order significant interaction terms that include the TS factor are $PT \times PS \times TS$, $PT \times ASTH \times TS$, and

$PS \times ASTH \times TS$. Since the most important experimental factor of interest is the TS factor, further analysis is performed to identify any differences among the six TS heuristics over the effect slices of these interaction terms. As presented in Table 8.23, the TS heuristics are significantly different when ASTH is 180 seconds with both large and small/medium size problems. Also, the difference is significant when ASTH is 180 seconds for both type 1 and type 2 problems.

Table 8.23. Tests of Slice Effects for Differences Among the Levels of TS (Makespan)

Effect	PS†	PT†	ASTH†	Ndf ‡	Ddf ‡	F–value	p–value
$PS \times PT \times TS$	Large	Type 1		5	120	4.47	0.0009
	Large	Type 2		5	120	24.18	<0.0001
	Small/Med.	Type 1		5	120	5.99	<0.0001
	Small/Med.	Type 2		5	120	3.29	0.0800
$PS \times ASTH \times TS$	Large		30	5	239	0.93	0.4628
	Large		180	5	239	38.32	<0.0001
	Small/Med.		30	5	239	1.65	0.1467
	Small/Med.		180	5	239	9.09	<0.0001
$PT \times ASTH \times TS$		Type 1	30	5	239	1.45	0.2064
		Type 1	180	5	239	12.07	<0.0001
		Type 2	30	5	239	1.10	0.3593
		Type 2	180	5	239	34.72	<0.0001

† *PT*: Problem Type, *PS*: Problem Size (large, small/medium), ASTH: Avg. setup time per feeder on HSPM

‡ Ndf: Numerator degrees of freedom, Ddf: Denominator degrees of freedom

Since significant differences are found, *multiple comparisons*, using Tukey's studentized range test, are performed for the combinations of the problem size (PS) and problem type (PT) factors. Table 8.24 presents the estimates of the percentage deviations of the TS heuristics for all four combinations of these factors.

Table 8.24. Percentage Deviations and Homogenous Groups of TS Heuristics for Different Levels of PS, PT, and ASTH Factors (Makespan)

PS	PT	TS1	TS2	TS3	TS4	TS5	TS6
Large	Type1	• 9.69	• 8.81	• • 9.06	• 8.88	• 8.81	• 8.78
Small/Med.	Type1	• 2.87	• 2.00	• • 2.25	• 1.85	• 1.99	• 1.83
Large	Type2	• 8.23	• 6.37	• 7.54	• 6.59	• 6.37	• 6.37
Small/Med.	Type2	• 2.01	• 1.35	• 1.84	• 1.42	• 1.35	• 1.35
PS	ASTH	TS1	TS2	TS3	TS4	TS5	TS6
Large	30	• 6.87	• 6.51	• 6.68	• 6.52	• 6.5	• 6.5
Large	30	• 11	• 8.67	• 9.92	• 8.94	• 8.67	• 8.65
Small/Med.	180	• 1.94	• 1.49	• 1.73	• 1.44	• 1.49	• 1.49
Small/Med.	180	• 2.93	• 1.86	• • 2.35	• 1.83	• 1.86	• 1.71
PT	ASTH	TS1	TS2	TS3	TS4	TS5	TS6
Type 1	30	• 5.34	• 4.89	• 5.09	• 4.9	• 4.89	• 4.89
Type 1	180	• 7.18	• 5.92	• 6.22	• 5.82	• 5.92	• 5.73
Type 2	30	• 3.47	• 3.11	• 3.33	• 3.06	• 3.12	• 3.12
Type 2	180	• 6.75	• 4.62	• 6.05	• 4.94	• 4.62	• 4.62

† TS heuristics with a • mark in the same row are statistically not different.

In Table 8.24, the TS heuristics with a '•' mark in the same row are statistically not different. For example, for large size problems of type 1, TS2–TS6 cannot be statistically distinguished. In this case, TS1 and TS3 cannot be distinguished, either. In general, significant differences among the TS heuristics are observed as the problem size increases. For instance, there are profound differences for large size and type 1 problems. However, the number of groups in the problems (i.e., type of the problems) seem to affect the percentage deviation even for small/medium size problems. This is most likely due to the fact that the quality of the lower bound from the column generation/B&P algorithm is better when the number of board types per board group is small, since the subproblems in such a case can be solved much quicker. In addition, as the problem size increases, the number of possible sequences increases and it becomes less likely to identify better solutions with tabu search if the advanced features such as the variable tabu list size and/or long term memory components are not used.

In summary, as in the case of minimizing the total flow time, TS1 and TS3 are consistently dominated by the other TS heuristics. This indicates that using variable tabu list size and long–term memory based on LTM–min to diversify the search have a strong impact on the quality of the solutions obtained by the TS heuristics. The differences between the TS heuristics are pronounced more for large size (and type 1) problems for the same reasons discussed for minimizing the total flow time.

The average computation time of each TS heuristic for each combination of the other experimental factors is presented in Table 8.25. As in the case of minimizing the total flow time, the computation times are higher for large size problems. As the number of groups increase, so does the number of possible sequences, which are generated, evaluated, and compared by the heuristics. There

Table 8.25. Average Computation Times (in seconds) of TS Solutions (Makespan)

ASTH	NM†	Size‡	Type	TS1	TS2	TS3	TS4	TS5	TS6
180	2	S/M	1	0.14	5.62	1.71	2.73	7.02	6.98
			2	0.02	0.55	0.14	0.26	0.84	0.88
		L	1	0.66	20.47	8.86	21.75	40.31	41.79
			2	0.23	5.16	0.93	2.04	6.43	6.42
	3	S/M	1	0.14	6.61	3.43	5.08	8.33	8.80
			2	0.06	0.45	0.16	0.23	0.89	0.95
		L	1	0.50	18.11	7.61	22.73	31.33	32.02
			2	0.28	5.60	1.02	3.34	7.16	7.17
30	2	S/M	1	0.10	4.79	1.06	2.06	6.83	7.20
			2	0.04	0.45	0.15	0.25	0.76	0.80
		L	1	0.58	23.02	3.47	36.90	54.24	55.04
			2	0.13	4.71	0.98	2.41	6.44	6.52
	3	S/M	1	0.25	11.81	1.85	5.12	20.55	22.13
			2	0.06	0.46	0.21	0.18	0.84	0.85
		L	1	0.86	20.19	7.02	19.16	40.94	42.28
			2	0.29	5.55	1.70	2.75	7.99	8.26
Overall Average				0.27	8.35	2.52	7.94	15.06	15.51

† NM: The number of machines

‡ S/M: small/medium size, L: large size

is a clear difference between the computation times for type 1 and type 2 problems. Since there are more individual board types within groups in type 1 problems, inside tabu search (that tries to find better sequences of individual board types within the groups) takes longer for type 1 problems. In general, the computation time of the heuristics are a little bit higher for three–machine problems than for two–machine problems. For the same number of board groups and board types within groups, the number of possible sequences are the same implying that

the heuristics have similar power to search the solution space of both problems. However, each sequence has to be evaluated over three machines rather than two, which increases the computation time.

Since, in general, there is not a statistically significant difference in the percentage deviations of TS2–TS6, TS3 seems to be the best choice since it has the minimal computation time. However, TS3 was significantly inferior than TS2, TS4, TS5, and TS6 for certain problem structures such as large size problems of type 2 (refer to Figure 8.24). As observed in the analysis of the experiments, TS2, TS4, TS5, and TS6 cannot be statistically distinguished for any of the problem structures. Therefore, it is concluded that TS4, with a smaller average computation time, is the TS heuristic that should be employed for the kinds of scheduling problems addressed here.

The other most important experimental factor of interest is the ASTH factor, which significantly affects the percentage deviation of the TS heuristics. Over all problem structures, the percentage deviations are estimated to be 4.10% and 5.70% when ASTH is 30 and 180 seconds, respectively. In general, the percentage deviation of the TS heuristics seem to increase as ASTH is changed from 30 seconds to 180 seconds.

Observe in the ANOVA table that the significant interaction terms that include ASTH are the $PS \times ASTH$, $PS \times ASTH \times TS$, and $PT \times ASTH \times TS$ terms. The last two terms are not much of an interest since the primary interest is to identify any differences among the TS heuristics, if possible, over the levels of the other factors, not the other way around. Therefore, additional insight is provided by the tests performed to identify any differences between the levels of the ASTH factor on the percentage deviation of the TS heuristics over the effect slices of the $PS \times ASTH$ interaction. Table 8.26 presents the estimates

of the percentage deviations of the TS heuristics for the two levels of ASTH separately for each slice (for each problem size). As presented in the same table, the differences are significant only when large size problems are considered. As in the case of minimizing the makespan, this may be because of the fact that the range of the completion times of the new columns to be generated during the column generation/B&P algorithm depends on the magnitudes of the setup and run times on the machines, and the number of columns need to be generated increase as the range of the completion times increase. Most importantly, the LP–relaxation of MILP1 is much tighter when ASTH is 30 seconds than it is 180 seconds (refer to Figure 8.8 on page 203). Since the proposed lower bounds are as good as the LP–relaxation values of MILP1, they are much better when ASTH is 30 seconds.

Table 8.26. Tests of Slice Effects for Differences Among the Levels of ASTH (Makespan)

PS	ASTH	
	30	180
Large	• 6.60	• 9.31
Small/Medium	• 1.60	• 2.09

Also observe that the percentage deviations are better for minimizing the total flow time than for minimizing the makespan, but the difference is very small. First, the TS heuristics terminate the search when the number of iterations without improvement reaches a predefined limit. When minimizing the makespan, different but similar sequences of board types may have the same makespan value,

since, for example, swapping two adjacent board types in the same board group can still produce the same makespan for the new sequence. Such a change would immediately change the mean/total flow time of such a sequence. Therefore, in the case minimizing the makespan, the TS heuristics may actually be producing different sequences but the best solution found may not be improved much and the heuristics may terminate earlier, even though the limit on the number of iterations without improvement was determined appropriately. Note that, especially TS1 has inferior percentage deviations from the lower bounds since it is the simplest TS heuristic. The impacts of variable tabu list sizes and long–term memory components in the TS heuristics seem to be pronounced more due to the same reason. Second, recall from the previous chapter that in the case of minimizing the makespan, the individual board types within board groups do not necessarily follow the shortest processing rule in an optimal solution to the last subproblem, SP(m), even when the dual variables are appropriately bounded between 1 and 0. Hence, SP(m) generally takes longer to solve and, in such cases, the lower bound could not be improved a lot within the five–hour time limit. However, most of the time, SP(m) could be skipped and tight lower bounds are obtained. Most importantly, the LP–relaxation of MILP1 is generally tighter with the objective of minimizing the makespan than minimizing the total flow time (this issue is discussed in detail in the next section). Since the proposed lower bounds are as good as the LP–relaxation value of MILP1, they are better with the objective of minimizing the makespan. On the average, this alleviates the poor approximation due to the difficulty in solving the last subproblem and the percentage deviations for minimizing the total flow time and the makespan are not much different.

8.4.4. MILP1 and MILP3 Performance

The problems generated for the experiments are attempted to be solved with MILP1 and MILP3 formulations using state–of–the-art optimization package CPLEX. All the small/medium and large size problems have been tried to be solved with MILP1. However, as mentioned earlier, MILP3 cannot be used to solve large size problems because of limited computing memory. Hence, only small/medium size problems have been attempted with MILP3. As anticipated, large size problems, and even some of the small/medium size problems could not be solved optimally with either of the two formulations within the five–hour limit. Detailed results regarding the performance of MILP1 and MILP3 on solving the problems are presented in Appendix D.4.

Tables 8.27 and 8.28 below summarize the number of problems attempted with MILP1 and MILP3, respectively, the number of problems that could be solved optimally, the number of problems for which a feasible solution could be identified but its optimality could not be proven within five hours, and the number of problems for which no feasible solution could be identified within the five–hour limit. The percentages of these figures are also presented.

Majority of the small/medium size problems could be solved optimally for both measures of performance by MILP1 and MILP3. MILP1 was able to optimally solve 65.6% of the small/medium size total flow time minimization problems, and 62.5% of the small/medium size makespan minimization problems. However, MILP1 was not able to identify even a feasible solution to one-fourth of such problems. On the other hand, the performance of MILP1 is remarkably inferior for large size problems. None of the large size problems could be optimally solved within the five–hour limit for both minimizing the makespan and the total

Table 8.27. Performance of MILP1 on Solving Small/Medium and Large Size Problems

Size	ASTH	Total Flow Time				Makespan			
		Number of Problems				Number of Problems			
		Total	Optimal	Feasible	Infeas.‡	Total	Optimal	Feasible	Infeas.‡
S/M†	180	16	11	1	4	16	10	3	3
	30	16	10	2	4	16	10	2	4
Total		32	21	3	8	32	20	5	7
Percentage		100	65.6	9.4	25.0	100	62.5	15.6	21.9
L†	180	16	0	4	12	16	0	3	13
	30	16	0	6	10	16	0	3	13
Total		32	0	10	22	32	0	6	26
Percentage		100	0.0	31.3	68.7	100	0.0	18.8	81.3

† S/M: small/medium size, L: large size

‡ Infeasible: Number of problems found infeasible within five hours.

flow time. MILP1 could not identify a feasible solution for the majority of the problems. For minimizing the total flow time, MILP1 identified a feasible solution to only 31.3% of the problems; the same figure is only 18.8% when the objective was minimizing the makespan.

Similarly, MILP3 was able to optimally solve 68.8% of the small/medium size total flow time minimization problems. Contrastingly, though, 93.8% of the small/medium size makespan minimization problems have been optimally solved with MILP3. In addition, small/medium problems could be solved quicker with MILP3 than MILP1. Hence, for solving small/medium problems, MILP3 is a much better choice than MILP1, since it can solve more of the problems optimally and it is quicker than MILP1.

Table 8.28. Performance of MILP3 on Solving Small/Medium Size Problems

Size	ASTH	Total Flow Time				Makespan			
		Number of Problems				Number of Problems			
		Total	Optimal	Feasible	Infeas.‡	Total	Optimal	Feasible	Infeas.‡
S/M†	180	16	11	5	0	16	15	1	0
	30	16	11	5	0	16	15	1	0
Total		32	22	10	0	32	30	2	0
Percentage		100	68.8	31.3	0.0	100	93.8	6.3	0.0

† S/M: small/medium size

‡ Infeasible: Number of problems found infeasible within five hours.

The fact that the large size –even some of the small/medium size– problems cannot be solved with either of the MILP formulations is the very reason that TS heuristics are proposed to solve the scheduling problems addressed in this dissertation effectively and in a timely manner. Since the proposed TS heuristics provide extremely good quality solutions quickly, they can be used as powerful and flexible tools to aid in the scheduling decisions in electronics manufacturing companies.

For each problem for which a feasible solution could be identified (but its optimality could not be verified), the detailed results in Appendix D.2 also include the percentage gap (percentage deviation) of the feasible solution value and the best global lower bound at the end of the five–hour limit. The percentage deviation reported by CPLEX is quite inferior to the percentage deviation obtained by the proposed lower bounding methods, although both were allowed to run for the same amount of time. This also establishes the strength of the proposed B&P algorithms over CPLEX for solving the types of problems addressed in this dissertation.

Previously in section 4.4, the proposed formulations MILP1, MILP2, and MILP3 were compared in terms of the number of variables and constraints they possess, their ability to solve problems of various sizes, and the strength of their LP–relaxations. As mentioned in that section, both MILP2 and MILP3 try to assign each board group to a slot, while the individual board types within each group are scheduled by the constraint sets (4.2.8) and (4.2.9) in MILP2 and (4.3.8) and (4.3.9) in MILP3. These constraint sets are not binding for an optimal solution. In general, this loosens the LP–relaxation of the formulations. However, the fact that a binary variable is defined for each possible group sequence strengthens the LP–relaxation of MILP3. Hence, the LP–relaxation of MILP3 is in general much tighter than that of MILP2. MILP1, on the other hand, assigns each individual board type to a slot while making sure that the board types within the same group are sequenced contiguously, one after the other. Hence, MILP1 does not have any non–binding constraint set to loosen its LP–relaxation. Therefore, an interesting observation of the experimental results would regard to comparing the LP–relaxations of MILP1 and MILP3. Since MILP1 and MILP3 are based on two different modelling frameworks proposed in this dissertation for formulating group–scheduling problems with different setup time structures such as sequence–independent or sequence–dependent setup times with or without carryover, or with no setups at all, this comparison is worth making to provide a guide for future research on group–scheduling problems. For small/medium size problems (both two– and three–machine problems and both levels of ASTH), Figures 8.6 and 8.7 compare the LP–relaxations of MILP1 and MILP3 formulations for minimizing the total flow time and the makespan, respectively.

With the objective of minimizing the total flow time, the LP–relaxation of one formulation seems to be tighter than that of the other for certain types of

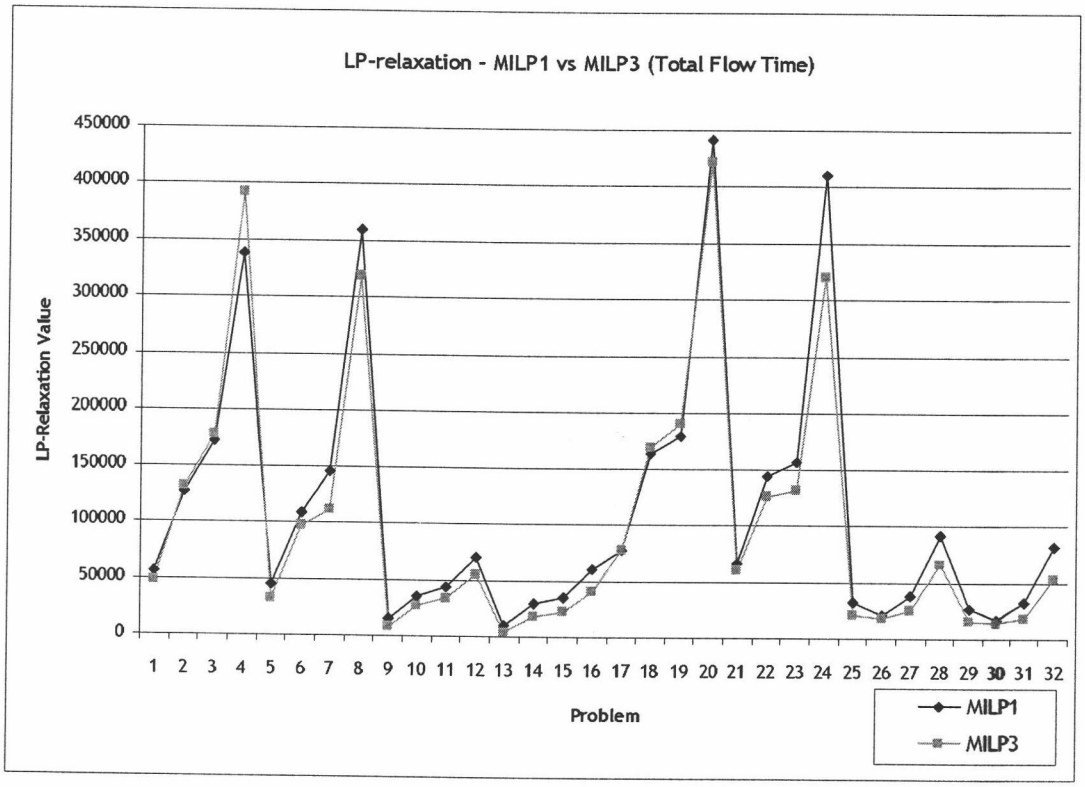

Figure 8.6. Comparison of LP–Relaxations of MILP1 and MILP3 (Total Flow Time)

problems. However, over all 32 problems, ANOVA revealed that the LP–relaxation of MILP1 is tighter than that of MILP3. The null hypothesis, which states that the mean of the difference of the LP–relaxation values is zero, is rejected with a p–value of 0.0052. (The mean of the LP–relaxation values of MILP1 minus the LP–relaxation values of MILP3 is 12122.6.)

Similarly, with the objective of minimizing the makespan, the LP–relaxation of one formulation seems to be tighter than that of the other for certain types of problems. Over all 32 problems, ANOVA revealed that the LP–relaxation of MILP3 is tighter than that of MILP1 (p–value of 0.002).

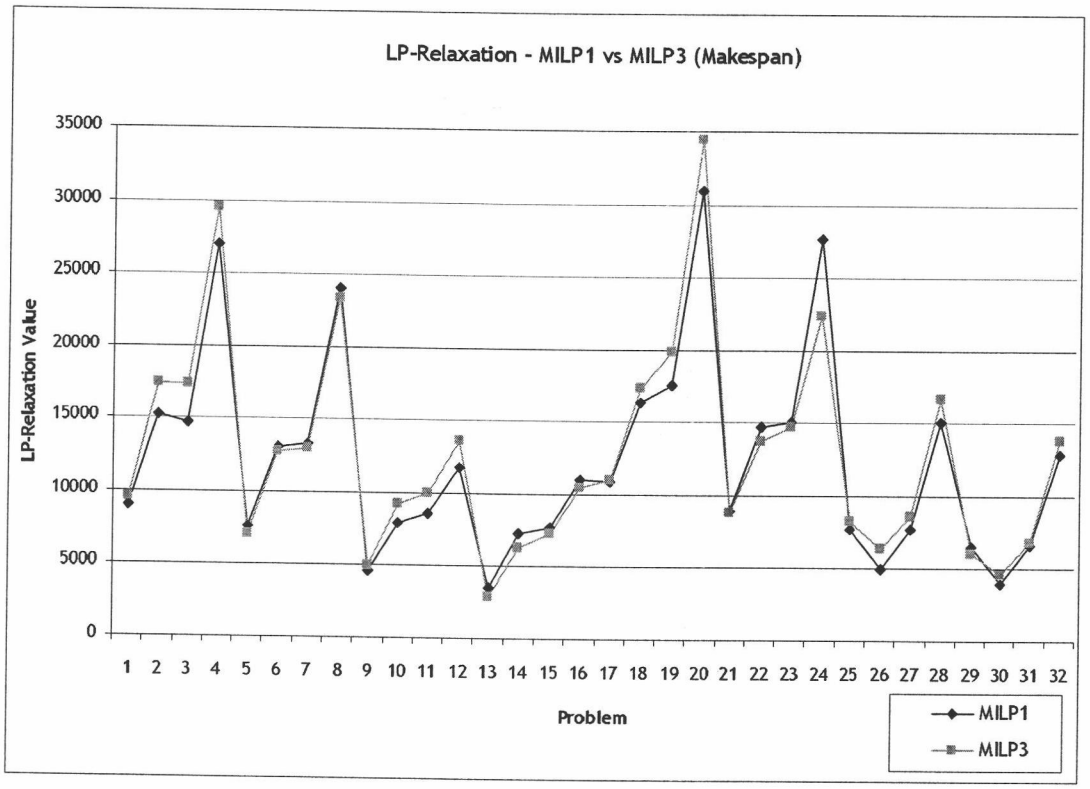

Figure 8.7. Comparison of LP–Relaxations of MILP1 and MILP3 (Makespan)

ANOVA did not reveal any statistically significant difference in the times required to solve the LP–relaxation values of MILP1 and MILP3 formulations for both measures of performance. Hence, for minimizing the sum of the flow times, it is best to use a MILP formulation similar to MILP1 rather than MILP3 with a heuristic or exact algorithm that is based on the LP–relaxation of a MILP formulation of a group–scheduling problem. similarly, it might be the best to use MILP3 for minimizing the makespan. However, it should be noted that MILP3 cannot handle large–size problems due to memory problems.

It has been stated previously that the LP–relaxation of MILP1 is much tighter when the average setup time per feeder (ASTH) is 30 seconds instead of

180 seconds. This is somewhat intuitive since when the setup times are smaller, the length of a schedule becomes smaller, reducing any numerical differences between the integer optimal solution and the LP–relaxation value. The tightness of an LP–relaxation can be considered as the closeness of the LP–relaxation value to the actual optimal solution. In other words, one would desire a formulation for which the percentage deviation of the optimal solution from the LP–relaxation value is as small as possible. For the problems generated in the experiments, Figure 8.8 depicts the percentage deviations of the solutions identified by TS6 from the LP–relaxation values of MILP1 for the two levels of the ASTH factor. Ideally, the percentage deviations would be compared with the actual optimal solutions. However, the optimal solutions are not available due to the complexity inherent in the problems. Instead, the solution identified by TS6 are used as an upper bound. Note that, TS6 is numerically the best TS heuristic and the quality of its solutions are extremely good as presented in the previous section.

The average percentage deviation for each case is shown as a straight line in Figure 8.8. Observe that, for minimizing the total flow time, the LP–relaxation is much tighter when ASTH is 30 seconds, since the percentage deviation with respect to the LP–relaxation value is lower than when ASTH is 180 seconds (the average percentage deviation is almost half). The same observation applies to the case with minimizing the makespan, as well. Also notice that the LP–relaxation values are closer to the upper bound for minimizing the makespan than for minimizing the total flow time (the deviations are smaller when minimizing the makespan). Since the column generation/B&P bound is as good as the LP–relaxation value, better lower bounds could be obtained when ASTH was 30 seconds. Also the lower bounds were slightly better for minimizing the makespan

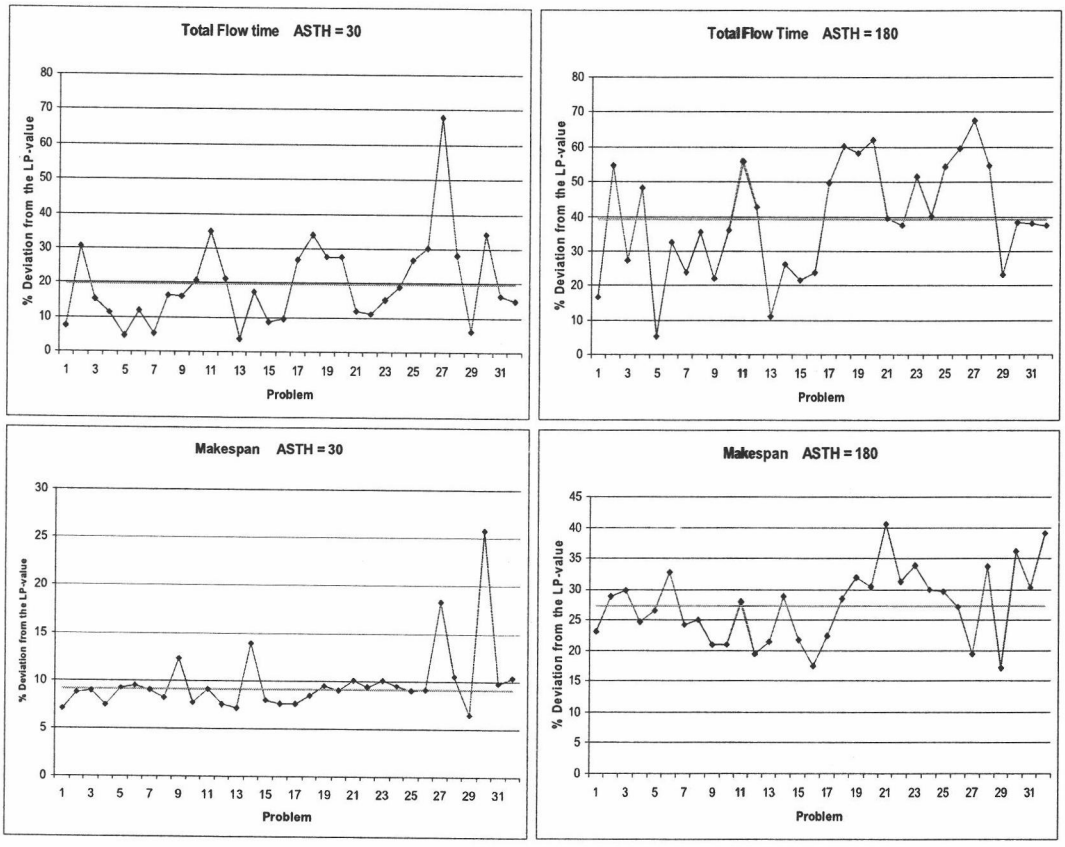

Figure 8.8. Comparison of the LP–Relaxation Values of MILP1 over the Two Levels of ASTH

than for minimizing the total flow time for the same reason. This explains why the percentage deviations of the proposed TS heuristics are better in such cases.

8.4.5. A Large Size Real Industry Problem

This section demonstrates the applicability of the algorithms proposed in this dissertation to solve a large size problem obtained from a PCB manufacturing company. The data involving the problem is presented in Appendix E.

The problem consists of a total of 13 board types. Three of the board types are grouped together to form a board group, while each of the remaining 10 board types stands as its own group. Hence, the problem is similar to a large size type 2 problem generated in the computational experiments. The PCB manufacturing company that provided the data implemented a two–machine assembly system that is similar to the one presented in Figure 3.1 on page 31. Machine 1 is a Fuji CP6 high–speed chip shooter (a high–speed placement machine), while Machine 2 is a Fuji IP3 flexible pick and place machine (a multi–function placement machine). The average setup time per feeder on both machines is 8 seconds per feeder. Understandably, the characteristics of these machines are slightly different from the ones considered previously in this dissertation; hence the difference in the average setup times per feeder values. However, the two– and three–machine assembly systems considered in this dissertation are typical to PCB assembly processes. All the algorithms proposed in this dissertation are applicable to *any and all* PCB assembly systems, as long as the component–feeder assignments of the board types, the run times of each of these board types on each machine (usually computed as the cumulative time it takes to insert all the components required of a board on the machine), and the average setup time per feeder on each machine are provided.

Unfortunately, the company that provided the data could not provide any information about the makespan/total flow time that would be observed for this particular data at their production line with the scheduling method currently in use. Typically, manufacturing companies do not keep track of the completion times of their products; hence, no performance comparison could be made. Nevertheless, it is important to note that the proposed algorithms can easily be applied

to different assembly systems, and large size problems encountered in the industry can be efficiently solved.

Table 8.29 presents the computation times of the six TS heuristics for the large size real industry problem.

Table 8.29. Computation Times (in seconds) of the TS Heuristics for the Large Size Real Industry Problem

TS Heuristic	Total Flow Time	Makespan
TS1	0.156	0.093
TS2	2.421	0.843
TS3	0.421	0.265
TS4	0.703	0.359
TS5	3.281	1.187
TS6	3.621	1.187

For minimizing the total flow time, the column generation/B&P algorithm identified a lower bound of 48456 in 5 hours. Table 8.30 presents the solutions identified by each of the six TS heuristics and their percentage deviations from this lower bound.

As observed, TS2, TS4, TS5, and TS6 all identified the same sequence of board groups and board types with a total flow time of 51459. The percentage deviation of this solution from the lower bound is 6.20%. In the absence of any results from the company for benchmarking and to illustrate how the total flow time values can be very different even for similar sequences, the table also includes *five best* out of a thousand randomly generated sequences and their total flow times. The random sequences are, as expected, highly inferior to the sequences identified

Table 8.30. Different Solutions for the Large Size Real Industry Problem (Total Flow Time)

Method	Sequence	TFT	% Dev.
TS1	G_{11}-G_9-G_5-G_1-G_7-G_2-G_{10}-(G_{41}-G_{42}-G_{43})-G_8-G_6-G_3	53291	9.98
TS2	G_2-G_5-G_{11}-G_9-G_7-G_1-G_{10}-(G_{41}-G_{42}-G_{43})-G_8-G_6-G_3	51459	6.20
TS3	G_1-G_7-G_{11}-G_9-G_5-G_2-G_{10}-(G_{41}-G_{42}-G_{43})-G_8-G_6-G_3	52918	9.21
TS4	G_2-G_5-G_{11}-G_9-G_7-G_1-G_{10}-(G_{41}-G_{42}-G_{43})-G_8-G_6-G_3	51459	6.20
TS5	G_2-G_5-G_{11}-G_9-G_7-G_1-G_{10}-(G_{41}-G_{42}-G_{43})-G_8-G_6-G_3	51459	6.20
TS6	G_2-G_5-G_{11}-G_9-G_7-G_1-G_{10}-(G_{41}-G_{42}-G_{43})-G_8-G_6-G_3	51459	6.20
Random	G_2-G_9-G_{11}-G_7-G_1-G_6-G_5-G_8-G_{10}-(G_{43}-G_{41}-G_{42})-G_3	55179	13.87
Random	G_5-G_1-G_7-G_2-G_{11}-G_9-(G_{41}-G_{42}-G_{43})-G_{10}-G_8-G_6-G_3	55713	14.98
Random	G_2-G_7-G_{11}-G_6-G_5-G_9-G_{10}-(G_{43}-G_{41}-G_{42})-G_1-G_8-G_3	57382	18.42
Random	G_9-G_{10}-G_5-G_{11}-G_1-G_7-G_2-(G_{41}-G_{42}-G_{43})-G_8-G_6-G_3	58111	19.92
Random	G_1-G_6-G_5-G_7-G_{11}-G_2-G_9-G_8-G_{10}-(G_{43}-G_{42}-G_{41})-G_3	59045	21.85

by the proposed tabu search heuristics. An experienced practitioner in industry would be expected to identify better sequences. However, it is not expected that the quality of the solutions suggested even by an experienced practitioner would be close to the quality of the TS heuristic solutions.

For minimizing the makespan, the lower bound identified by the column generation/B&P algorithm is 9350 in 1895 seconds (approximately 32 minutes). Table 8.31 presents the solutions identified by each of the six TS heuristics and their percentage deviations from this lower bound.

As observed, TS2, TS4, TS5, and TS6 all identified the sequence of board groups and board types with a makespan of 9516. The percentage deviation of this solution from the lower bound is 1.78%. The table also includes *five best* out of a thousand randomly generated sequences and their makespan values. Again,

Table 8.31. Different Solutions for the Large Size Real Industry Problem (Makespan)

Method	Sequence	Makespan	% Dev.
TS1	G_2-G_1-G_{11}-G_9-G_6-G_3-(G_{43}-G_{42}-G_{41})-G_7-G_{10}-G_5-G_8	9516	1.78
TS2	G_2-G_1-G_{11}-G_9-G_6-G_3-(G_{43}-G_{42}-G_{41})-G_7-G_{10}-G_5-G_8	9516	1.78
TS3	G_2-G_1-G_{11}-G_9-G_6-G_3-(G_{43}-G_{42}-G_{41})-G_7-G_{10}-G_5-G_8	9516	1.78
TS4	G_2-G_1-G_{11}-G_9-G_6-G_3-(G_{43}-G_{42}-G_{41})-G_7-G_{10}-G_5-G_8	9516	1.78
TS5	G_2-G_1-G_{11}-G_9-G_6-G_3-(G_{43}-G_{42}-G_{41})-G_7-G_{10}-G_5-G_8	9516	1.78
TS6	G_2-G_1-G_{11}-G_9-G_6-G_3-(G_{43}-G_{42}-G_{41})-G_7-G_{10}-G_5-G_8	9516	1.78
Random	G_2-G_9-G_3-G_8-G_6-G_{11}-G_7-G_{10}-G_5-G_1-(G_{42}-G_{41}-G_{43})	9604	2.72
Random	G_2-G_5-G_6-(G_{42}-G_{43}-G_{41})-G_9-G_3-G_{10}-G_{11}-G_8-G_7-G_1	9657	3.28
Random	G_1-G_7-G_9-G_{11}-G_2-(G_{42}-G_{41}-G_{43})-G_6-G_3-G_5-G_8-G_{10}	9669	3.41
Random	G_2-G_{11}-G_3-G_1-G_7-G_8-G_5-G_6-G_9-G_{10}-(G_{41}-G_{42}-G_{43})	9672	3.44
Random	G_2-G_3-G_5-G_1-G_7-G_{10}-G_8-(G_{43}-G_{41}-G_{42})-G_6-G_9-G_{11}	9683	3.56

the random sequences are, as expected, highly inferior to the sequences identified by the proposed tabu search heuristics. Even an experienced practitioner is not expected to suggest sequences with such good qualities.

The results presented here for a large size real industry problem, combined with the quality of the tabu search heuristic solutions in the experiment performed in this dissertation, strongly suggests the applicability and the high performance of the proposed algorithms for a wide range of different problem structures in terms of different problem sizes (small/large), problem types (high–mix low–volume or low–mix high–volume), assembly systems (two– and three–machine) and different setup times per feeder on the machines.

9. CONCLUSIONS

This dissertation addressed the "multi–machine carryover sequence–dependent group–scheduling problem with anticipatory setups," which typically arises in the production of printed circuit boards (PCBs). In PCB manufacturing different board types requiring similar components are grouped together using group technology principles in order to reduce the setup times and increase throughput. Quite distinct from the traditional machine scheduling problems involving setup times, the setup times required of a board group depends not just on the immediately preceding board group, but on *all* of the preceding groups and the *order* they are processed. Therefore, the challenges encountered in this research are far greater and distinctly different from those reported previously. All the previous research efforts reported in the published research avoids the carryover structure of the setup times by simplifying the problem with myopic approaches. This results in losing valuable information about the actual problem and leads to inferior solutions. This dissertation, on the other hand, addresses the problem *with* the carryover setup times structure and develops algorithms capable of identifying high quality solutions.

The scheduling decisions involve determining the sequence of board groups as well as the sequence of individual board types within groups so as to minimize appropriate measures of performance. The focus was on two separate objectives, namely, minimizing the makespan and minimizing the mean (total) flow time. The problem with each of the objectives is shown to be $\mathcal{NP}$–hard in the strong sense. Therefore, the primary interest was to develop algorithms to solve these challenging problems effectively in a timely manner to help the electronics manufacturing companies improve their competitiveness. Consequently, high level

metasearch heuristic algorithms based on the concept known as *tabu search* are developed. Advanced features of tabu search, such as the long–term memory in order to intensify/diversify the search and variable tabu–list sizes, are utilized in the proposed heuristics. Incorporating different advanced features resulted in a total of six tabu search heuristic algorithms, namely TS1, TS2, TS3, TS4, TS5, and TS6. Apart from optimizing regular daily scheduling problems, the proposed tabu search heuristics provide a great deal of flexibility to the personnel in charge of scheduling the PCB assembly. The proposed tabu search based heuristics provide a basis for a flexible decision support system to easily cope with situations when a major change occurs, such as product mix changes that happen frequently in electronics manufacturing, urgent prototype series production requests, temporary lack of the required components, and machine breakdowns.

The impact of the advanced features of tabu search is found to significantly improve the solution quality. Specifically, TS1, which uses fixed tabu list size and no long term memory, is found to be significantly inferior to the other tabu search heuristics. In general there was not a significant improvement when long term memory based on LTM–max (for intensification purposes) was used with fixed or variable tabu list sizes. The benefit of employing variable tabu list sizes and long term memory based on LTM–max so as to diversify the search is found highly substantial. The benefits are pronounced more especially for large size problems. Using long term memory based on LTM–max only deepens the search within the regions visited previously, and as the solution space enlarges, it is likely that better solutions can be found at other regions of the solution space, which is highly effectively achieved by long term memory based on LTM–min. Also, variable tabu list sizes provide a powerful means to scan the solution space for better solutions. However, the additional computational effort demanded by

variable tabu list sizes did not pay since the difference between using variable tabu list sizes combined with LTM–max and using fixed tabu list sizes with LTM–max was not significant. For certain cases TS6, which utilize variable tabu list sizes and LTM–min, significantly outperformed TS4, which uses a fixed tabu list size and LTM–min. However, it is concluded that TS4 would be the best choice in general, since the computational effort demanded by TS4 is significantly smaller.

In contrast with many reported research efforts that compare different heuristic algorithms, this dissertation not only performs comparisons among different heuristics, but also *quantifies* the quality of the heuristic solutions. In the absence of knowing the actual optimal solutions, the challenge was to obtain tight lower bounds both on the optimal makespan and the optimal total flow time. The quality of a solution is then quantified as its percentage deviation from the lower bound. Based on the minimum possible setup times, this research proposed a lower bounding procedure for the *two–machine* problem, which is called procedure *Minsetup*, and is capable of identifying tight lower bounds. However, the lower bounds from the procedure Minsetup are not as close to the actual optimal solutions as desired to reflect the actual performance of the proposed tabu search heuristics.

Tighter lower bounds are identified using a mathematical programming decomposition approach. First, two novel modeling frameworks are proposed. These modelling frameworks can be used to formulate a mathematical programming model for any group–scheduling problem. Utilizing one of the mathematical formulations, a novel *branch–and–price* (B&P) algorithm is developed for the group–scheduling problems addressed in this dissertation. Most of the column generation and B&P algorithms in the literature proposed for machine scheduling address parallel–machine problems. The proposed B&P algorithm, on the other

hand, is one of the few that are developed for the flowshop scheduling problems (with sequential machines) and, to the best of our knowledge, is the first B&P approach addressing group–scheduling problems. A Dantzig–Wolfe reformulation of the problem is constructed and a column generation algorithm is described to solve the linear programming relaxation of the master problem, where separate single–machine subproblems are designed to identify new columns if and when necessary. To enhance the efficiency of the algorithm, approximation algorithms are developed to solve the subproblems as well. Ways to obtain valid global lower bounds in each iteration of the column generation algorithm are shown, and effective branching rules are proposed to partition the solution space of the problem at a node where the solution is fractional. These branching rules are incorporated into the subproblems in an efficient way, as well. In order to alleviate the slow convergence of the column generation process, a stabilizing method is developed. The method, based on a primal–dual relationship, bounds the dual variable values at the beginning and gradually relaxes the bounds as the process marches towards the optimal dual variable values. Finally, several implementation issues, such as constructing a feasible initial master problem, column management, search strategy, and customized termination criteria are addressed.

The ability to asses the quality of the tabu search solutions was primarily as a result of identifying tight lower bounds. The proposed column generation/B&P algorithms were able to obtain quality lower bounds primarily because of (1) the effectiveness of the approximation algorithms developed for solving the subproblems, (2) the identification of certain cases for which some of the subproblems can be skipped without solving, and (3) the impact of the proposed stabilization methods.

A carefully designed computational experiment is performed to compare the proposed tabu search algorithms, to estimate the quality of the solutions they identify, and to determine problem parameters that significantly affect the quality of the tabu search solutions. For both *low-mix high-volume* and *high-mix low-volume* production environments, the results of the experiment confirm the high performance of tabu search algorithms in identifying extremely good quality solutions with respect to the proposed lower bounds. For minimizing the total (mean) flow time, the average percentage deviation of TS4 solutions is only 1.87% for small/medium size problems and 6.66% for large size problems. For minimizing the makespan, the average percentage deviation of TS4 solutions is only 1.64% for small/medium size problems and 7.75% for large size problems.

The proposed algorithms are applicable to a wide range of PCB assembly systems composed of different placement machines. As such, the algorithms are applied to a large size real industry problem. The proposed solutions are shown to be extremely better than randomly generated solutions, and even an experience practitioner in industry is not expected to suggest solutions with qualities close to the ones identified by the proposed tabu search algorithms. For minimizing the total (mean) flow time, the best tabu search heuristic was able to identify a solution that is 6.20% off of the lower bound. Similarly, the percentage deviation of the best tabu search solution from the lower bound for the case of minimizing the makespan is only 1.78%.

In sum, the quality of the tabu search heuristic solutions for randomly generated instances that are representative of industry problems, combined with the results presented for a large size real industry problem, strongly suggests the applicability and the high performance of the proposed algorithms for a wide range of different problem structures in terms of different problem sizes (small and

large), problem types (high–mix low–volume and low–mix high–volume), assembly systems (two– and three–machine) and different setup times per feeder on the machines.

As for future research, there are several potential areas. First, the procedure Minsetup based lower bounds are found to be highly inferior to the lower bounds provided by the column generation/B&P algorithm. That's why, procedure Minsetup based lower bounds are not analyzed in this dissertation. Another reason for not doing so is the fact that procedure Minsetup identifies lower bounds for the two–machine problems only. However, procedure Minsetup based lower bounds can be obtained for the three–machine problems by solving the converted problem(s) via mathematical programming. Note that, the algorithms due to Yoshida and Hitomi [122] and Ham et al. [58] are used to *optimally* solve corresponding problems for minimizing the makespan and minimizing the total flow time, respectively. The three machine problem with carryover sequence–dependent setup times can still be converted to one with sequence–independent setup times without carryover, and lower bounds can be obtained by solving the corresponding problems via mathematical programming. It is an easy task to modify MILP1, MILP2, or MILP3 formulations to represent sequence–independent setup times problems. Although such lower bounds are expected to be inferior to the lower bounds identified by the column generation/B&P algorithms, it is still worth pursuing research for additional insights.

Instead of solving the problem, the proposed column generation/B&P algorithms are used to identify tight lower bounds in this dissertation. The proposed B&P algorithm can be used to solve small size instances. However, it does not seem as a practical approach to solve such problems since the subproblems are hard to solve for large size problems. However, a B&P algorithm can be devel-

oped as a heuristic to obtain good quality solutions. It is possible to solve the subproblems *only* with tabu search (with tabu search column generator, TSCG), without solving the subproblem formulations exactly. When the subproblems are not solved optimally but approximately, the pricing scheme is said to be *partial pricing.* In this case, none of the nodes can be pruned in the B&P tree (unless an integer solution is identified) and the method resembles an enumeration search and is not likely to solve large size problems in a timely manner. A B&P tree with partial pricing can serve to identify heuristic solutions to the problem. It has been experienced that the column generation algorithm was able to identify primal solutions better than any one of the TS solutions, although this was observed highly rarely.

In addition, tabu search is a relatively new concept and it still continues to evolve. There are many different aspects of the concept, such as incorporating a random component into the algorithm, that can be implemented and tested for better results. There is a lot of room in designing an effective tabu search algorithm with different and creative means.

BIBLIOGRAPHY

[1] Ahmadi, J., Ahmadi, R., Matsuo, H., Tirupati, D., Component Fixture Partitioning/Sequencing for Printed Circuit Board Assembly with Concurrent Operations, *Operations Research*, **43**, 444–457, 1995.

[2] Ahmadi, J., Wurgaft, H., Design for Synchronized Flow Manufacturing, *Management Science*, **40**, 1469–1483, 1994.

[3] Ammons, J.C., Carlyle, M., Cranmer, L., Depuy, G.W., Ellis, K.P., McGinnis, L.F., Tovey, C.A., Xu, H., Component Allocation to Balance Workload in Printed Circuit Card Assembly Systems, *IIE Transactions*, **29(4)**, 265–276, 1997.

[4] Allahverdi, A., Gupta, J.N.D., Aldowaisian, T., A Review of Scheduling Research Involving Setup Considerations, *Omega–International Journal of Management Science*, **27**, 219–239, 1999.

[5] Allison, J.D., Combining Petrovs Heuristic and the CDS Heuristic in Group Scheduling Problems, *Proceedings of the 12th Annual Conference on Computers and Industrial Engineering*, Orlando, FL, 457–461, 1990.

[6] Al–Qattan, I., Designing GT Cells Enhanced by Group Scheduling: Key to Flexible Manufacturing System, *Proceedings of the IIE Integrated Systems Conference*, St. Louis, MO, 25–30, 1988.

[7] Amini, M.M., Barr, R.S.,Network Reoptimization Algorithms: A Statistically Designed Comparison, *ORSA Journal on Computing*, **5(4)**, 395–408, 1993.

[8] Amini, M.M., Racer, M.,A Rigorous Computational Comparison of Alternative Solution Methods for the Generalized Assignment Problem, *Management Science*, **40(7)**, 868–890, 1994.

[9] Askin, R.G., Dror, M., Vakharia, A. J., Printed Circuit Board Family Grouping and Component Allocation for a Multimachine Open Shop Assembly Cell, *Naval Research Logistics*, **41**, 587–608, 1994.

[10] Barnea, A., Sipper, D., Set-Up Reduction in PCB Automated Assembly, *Computer Integrated Manufacturing Systems*, **6(1)**, 18–26, 1993.

[11] Barnhart, C., Johnson, E., Newhauser, G., Savelsbergh, M., Vance, P., Branch–and–Price: Column Generation for Solving Huge Integer Programs, *Operations Research*, **46(3)**, 316–329, 1998.

[12] Barr, R.S., Golden B.L., Kelly, J.P., Resende, M.G.C., Stewart, W.R.,Designing an Reporting on Computational Experiments with Heuristic Methods, *Journal of Heuristics*, **1**, 9–32, 1995.

[13] Bhaskar, G., Narendran, T.T., Grouping PCBs for Set-Up Reduction: A Maximum Spanning Tree Approach, *International Journal of Production Research*, **34(3)**, 621–632, 1996.

[14] Baker, K.R., Heuristic Procedures for Scheduling Job Families with Setups and Due Dates, *Naval Research Logistics*, **46**, 978–991, 1999.

[15] Balakrishnan, A., Vanderbeck, F., A Tactical Planning Model for Mixed–Model Electronics Assembly Operations, *Operations Research*, **47(3)**, 395–409, 1999.

[16] Bard, J.F., Clayton, R.W., Feo, T.A., Machine Setup and Component Placement in Printed Circuit Board Assembly, *International Journal of Flexible Manufacturing Systems*, **6**, 5–31, 1994.

[17] Barnes, J.W., Laguna, M., Glover, F., An Overview of Tabu Search Approaches to Production Scheduling Problems, in Brown, D.E., Scherer, W.T., editors, *Intelligent Scheduling Systems*,Kluwer Academic Publishers, Boston, MA, 101–127, 1995.

[18] Bazaraa, M.S., Jarvis, J.J., Sherali, H.D., editors, *Linear Programming and Network Flows*, second edition, John Wiley and Sons, New York, 1990.

[19] Ben–Arieh, D., Dror, M., Part Assignment to Electronic Insertion Machines: Two Machine Case, *International Journal of Production Research*, **28(7)**, 1317–1327, 1990.

[20] Ben–Arieh, D., Maimon, O., Annealing Method for PCB Assembly Scheduling on Two Sequential Machines, *International Journal of Computer Integrated Manufacturing*, **5(6)**, 361–367, 1992.

[21] Brandeau, M.L., Billington, C.A., Design of Manufacturing Cells: Operation Assignment in Printed Circuit Board Manufacturing, *Journal of Intelligent Manufacturing*, **2**, 95–106, 1991.

[22] Bülbül, K., Kaminsky, P., Yano, C., Flow Shop Scheduling with Earliness, Tardiness, and Intermediate Inventory Holding Costs, *Naval Research Logistics*, **51**, 407–445, 2004.

[23] Carmon, T.F., Maimon, O.Z., Dar–el, E.M., Group Set–up for Printed Circuit Board Assembly, *International Journal of Production Research*, **27**, 1795–1810, 1989.

[24] Chen, Z.–L., Lee, C.–Y., Parallel Machine Scheduling with a Common Due Window, *European Journal of Operational Research*, **136(3)**, 512–527, 2002.

[25] Chen, Z.–L., Powell, W., Solving Parallel Machine Scheduling Problems by Column Generation, *INFORMS Journal on computing*, **11(1)**, 78–94, 1999.

[26] Chen, Z.–L., Powell, W., A column Generation Based Decomposition Algorithm for a Parallel Machine Just–in–Time Scheduling Problem, *European Journal of Operational Research*, **116(1)**, 220–232, 1999.

[27] Coffin, M., Saltzman, M.J.,Statistical Analysis of Computational Tests of Algorithms and Heuristics, *INFORMS Journal on computing*, **12(1)**, 24–44, 2000.

[28] Crama, Y., Flippo, O.E., van de Klundert, J.J., Spieksma, F.C.R., The Component Retrieval Problem in Printed Circuit Board Assembly, *International Journal of Flexible Manufacturing Systems*, **8**, 287–312, 1996.

[29] Crama, Y., Flippo, O.E., van de Klundert, J.J., Spieksma, F.C.R., The Assembly of Printed Circuit Boards: A Case with Multiple Machines and Multiple Board Types, *European Journal of Operational Research*, **98**, 457–472, 1997.

[30] Crama, Y., van de Klundert, J.J., Spieksma, F.C.R., Production Planning Problems in Printed Circuit Board Assembly, *Discrete Applied Mathematics*, **123**, 339–361, 2002.

[31] Cunningham, P., Browne, J., A LISP–Based Heuristic Scheduler for Automatic Insertion in Electronics Assembly, *International Journal of Production Research*, **24(6)**, 1395–1408, 1986.

[32] Dantzig, G.B., Wolfe, P., Decomposition Principle for Linear Programs, *Operations Research*, **8**, 101–111, 1960.

[33] De Werra, D., Hertz, A., Tabu Search Techiques: A Tutorial and Application to Neural Networks, *OR Spektrum*, **11**, 131–141, 1989.

[34] Desrochers, M., Desrosiers, J., Solomon, M., A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows, *Operations Research*, **40(2)**, 342–354, 1992.

[35] Desrosiers, J., Dumas, Y., Solomon, M.M., Soumis, F., Time Constrained Routing and Scheduling, in Ball, M.E., Magnanti, T.L., Monma, C., Nehauser, G.L., editors, *Handbook in Operations Research and Management Sciences*, 8. volume on Network routing, Elsevier, Amsterdam, The Netherlands, 35–139, 1995.

[36] Dikos, A., Nelson, P.C., Tirpak, T.M., Wang, W., Optimization of High–Mix Printed Circuit Card Assembly Using Genetic Algorithms, *Annals of Operations Research*, **75**, 303–324, 1997.

[37] Dillon, S., Jones, R., Hinde, C.J., Hunt, I., PCB Assembly Line Setup Optimization Using Component Commonality Matrices, *Journal of Electronics Manufacturing*, **82**, 77–87, 1998.

[38] Du Merle, O., Villeneuve, D., Desrosiers, J., Hansen, P., Stabilized Column Generation, *Discrete Mathematics*, **1941–3**, 229–237, 1999.

[39] Escudero, L.F., S3 sets, An Extension of the Beale–Tomlin Special Ordered Sets, *Mathematical Programming: Series B*, **42**, 113–123, 1988.

[40] Eom, D.H., Shin, H.J., Kwun, I.H., Shim, J.K., Kim, S.S., Scheduling Jobs on Parallel Machines with Sequence–Dependent Family Set–up Times, *International Journal of Advanced Manufacturing Technology*, **19(12)**, 926–932, 2002.

[41] Ford, L.R., Fulkerson, D.R., A Suggested Computation for Maximal Multi-commodity Network Flows, *Management Science*, **5**, 97–101, 1958.

[42] Francis, R.L., Hamacher, H.W., Lee, C.Y., Yeralan, S., Finding Placement Sequences and Bin Locations for Cartesian Robots, *IIE Transactions*, **26**, 47–59, 1994.

[43] Gelogullari, C.A., Logendran, R., A Column Generation Algorithm for the Carryover Sequence Dependent Group Scheduling Problems, *Proceedings of the IIE 2004 Annual Conference* (CD-ROM), May 15-19, Houston, TX, USA, 2004.

[44] Garey, M.R., Johnson, D.S., editors, *Computers and Intractability: A Guide to the Theory of NP–Completeness*, W.H. Freeman, New York, 1979.

[45] Garey, M.R., Johnson, D.S., Sethi, R.R., The Complexity of Flowshop and Jobshop Scheduling, *Mathematics of Operations Research*, **1**, 117–129, 1976.

[46] Greenberg, H.J., Computational Testing: Why, How and How Much?, *ORSA Journal on computing*, **2(1)**, 94–97, 1990.

[47] Glover, F., Heuristics for Integer Programming Using Surrogate Constraints, *Decision Sciences*, **8(1)**, 156–1166, 1977.

[48] Glover, F., Future Paths for Integer Programming and Links to Artificial Intelligence, *Computers and Operations Research*, **13(5)**, 533–549, 1986.

[49] Glover, F., Tabu Search – Part I, *ORSA Journal on Computing*, **1(3)**, 190–206, 1989.

[50] Glover, F., Tabu Search – Part II, *ORSA Journal on Computing*, **2(1)**, 4–32, 1990.

[51] Glover, F., Tabu–Search: A Tutorial, *Interfaces*, **20**, 79–94, 1990.

[52] Glover, F., Laguna, M., Tabu Search, in Reeves, C.R., editor, *Modern Heuristics for Combinatorial Optimization*, Blackwell Scientific Publications, Oxford, 1993.

[53] Glover, F., Laguna, M., editors, *Tabu Search*, Kluwer Academic Publishers, Boston, MA, 1997.

[54] Gupta, J.N.D., Darrow, W.P., The Two–Machine Sequence Dependent Flowshop Scheduling Problem, *European Journal of Operational Research*, **24**, 439–446, 1986.

[55] Gupta, J.N.D., Ho, J.C., Scheduling with Two Job Classes and Setup Times to Minimize the Number of Tardy Jobs, *International Journal of Production Economics*, **42(3)**, 205–216, 1996.

[56] Günther, H.O., Gronalt, M., Zeller, R., Job Sequencing and Component Set–Up on a Surface Mount Placement Machine, *Production Planning and Control*, **9(2)**, 201–211, 1998.

[57] Hansen, P., The Steepest Ascent Mildest Descent Heuristic for Combinatorial Programming, *Presented at the Congress on Numerical Methods in Combinatorial Optimization*, Capri, Italy, 1986.

[58] Ham, I., Hitomi, K., Yoshida, T., editors, *Group Technology*, Kluwer Academic Publishers, 1985.

[59] Hashiba, S., Chang, T.C., Heuristic and Simulated Annealing Approaches to PCB Assembly Setup Reduction, in Olling, G.J., Kimura, F., editors, *Human Aspects in Computer Integrated Manufacturing*, IFIP Transactions B–3, Elsevier Science Publishers, Amsterdam, The Netherlands, 76977, 1992.

[60] Häyrinen, T., Johnsson, M., Jahtella, T., Smed, J., Nevalainen, O., Scheduling Algorithms for Computer–Aided Line Balancing in Printed Circuit Board Assembly, *Production Planning and Control*, **11(5)**, 497–510, 2000.

[61] Heragu, S.S., Group Technology and Cellular Manufacturing, *IEEE Transactions on Systems, Management, and Cybernetics*, **24(2)**, 203–215, 1994.

[62] Hillier, M.S., Brandeau, M.L., Optimal Component Assignment and Board Grouping in Printed Circuit Board Manufacturing, *Operations Research*, **46(5)**, 675–689, 1998.

[63] Hoogeveen, J., Parallel Machine Scheduling with a Common Due Window, *European Journal of Operational Research*, **136(3)**, 512–527, 2002.

[64] Hooker, J.N., Needed: An Empirical Science of Algorithms, *Operations Research*, **42(3)**, 201–212, 1994.

[65] Hooker, J.N., Testing Heuristics: We Have It All Wrong, *Journal of Heuristics*, **1**, 33–42, 1996.

[66] Ilog Inc., *ILOG CPLEX 9.0 Reference Manual*, Paris, France, 2004.

[67] Ilog Inc., *ILOG Branch&Price&Cut and Shortest Path Optimizers Manual*, Paris, France, 2003.

[68] Ji, P., Wan, Y.F., Planning for Printed Circuit Board Assembly: The State–of–the–Art Review, *International Journal of Computer Applications in Technology*, **14(4/5/6)**, 136–144, 2001.

[69] Johnson, S.M., Optimal Two– and Three–Stage Production Schedules with Set–up Times Included, *Naval Research Logistics Quarterly*, **1**, 61–68, 1954.

[70] Johnson, M., Smed, J., Observations on PCB Assembly Optimization, *Electronic Packaging and Production*, **41(5)**, 38–42, 2001.

[71] Johri, P.K., A Heuristic Algorithm for Loading New Work on Circuit Pack Assembly Lines, *International Journal of Production Research*, **28(10)**, 1871–1883, 1990.

[72] Kanet, J., Sridharan, V., Scheduling with Inserted Idle Time: Problem Taxonomy and Literature Review, *Operations Research*, **48(1)**, 99–110, 2000.

[73] Kim, Y.D., Lim, H.G., Park, M.W., Search Heuristics for a Flowshop Scheduling Problem in a Printed Circuit Board Assembly Process, *European Journal of Operational Research*, **91**, 124–143, 1996.

[74] Lee, C.–Y., Chen, Z.–L., Scheduling Jobs and Maintenance Activities on Parallel Machines, *Naval Research Logistics*, **47(2)**, 145–165, 2000.

[75] Leon, V.J., Peters, B.A., Replanning and Analysis of Partial Setup Strategies in Printed Circuit Board Assembly Systems, *International Journal of Flexible Manufacturing Systems*, **8(4)**, 389–412, 1996.

[76] Leon, V.J., Peters, B.A., A Comparison of Setup Strategies for Printed Circuit Board Assembly, *Computers and Industrial Engineering*, **34(1)**, 219–234, 1998.

[77] Liaee, M.M., Emmons, H., Scheduling Families of Jobs with Setup Times, *International Journal of Production Economics*, **51**, 165–176, 1997.

[78] Liao, C., Chuang, C., Sequencing with Setup Time and Order Tardiness Trade–offs, *Naval Research Logistics*, **43(7)**, 971–984, 1996.

[79] Lin, B.W., Rardin, R.L., Controlled Experimental Design for Statistical Comparison of Integer Programming Algorithms, *Management Science*, **25(12)**, 1258–1271, 1980.

[80] Lin, F.R., Shaw, M.J., Locascio, A., Scheduling Printed Circuit Board Production Systems Using the Two–Level Scheduling Approach, *Journal of Manufacturing Systems*, **16(2)**, 129–149, 1997.

[81] Lofgren, C.B., McGinnis, L.F., Dynamic Scheduling for Flexible Printed Circuit Card Assembly, *Proceedings of the IEEE Conference on Systems, Management, and Cybernetics*, Atlanta, GA, 1294–1297, 1986.

[82] Lofgren, C.B., McGinnis, L.F., Towey, C.A., Routing Printed Circuit Cards Through an Assembly Cell, *Operations Research*, **39**, 992–1004, 1991.

[83] Logendran, R., Gelogullari, C.A., Sriskandarajah, C., Minimizing the Mean Flow Time in a Two–Machine Group–Scheduling Problem with Carryover Sequence Dependency, *Journal of Robotics and Computer Integrated Manufacturing*, **19(1–2)**, 21–33, 2003.

[84] Logendran, R., Mai, L., Talkington, D., Combined Heuristics for Bi–Level Group Scheduling Problems, *International Journal of Production Economics*, **38(2–3)**, 133–145, 1995.

[85] Logendran, R., Nudtasomboon, N., Minimizing the Makespan of a Group Scheduling Problem: A New Heuristic, *International Journal of Production Economics*, **22**, 217–230, 1991.

[86] Logendran, R., Sonthinen, A., A Tabu Search–Based Approach for Scheduling Job–Shop Type Flexible Manufacturing Systems, *Journal of the Operational Research Society*, **48**, 264–277, 1997.

[87] Logendran, R., Sriskandarajah, C., Two–Machine Group Scheduling Problem with Blocking and Anticipatory Setups, *European Journal of Operational Research*, **69(3)**, 467–481, 1993.

[88] Logendran, R., Subur, F., Unrelated Parallel Machine Scheduling with Job Splitting, *IIE Transactions*, **36**, 359–372, 2004.

[89] Maimon, O., Shtub, A., Grouping Methods for Printed Circuit Boards, *International Journal of Production Research*, **29(7)**, 1370–1390, 1991.

[90] Magyar, G., Johnson, M., Nevalainen, O., On Solving Single Machine Optimization Problems in Electronics Manufacturing, *Journal of Electronics Manufacturing*, **9(4)**, 249–267, 1999.

[91] Mangin, C.–H, Line Efficiency and Productivity Measures, *Surface Mount Technology*, **10**, 114–116, 1999.

[92] Manne, A.S., On the Job–Shop Scheduling Problem, *Operations Research*, **8(2)**, 219–223, 1960.

[93] McGinnis, L.F., Ammons, J.C., Carlyle, M., Cranmer, L., Depuy, G.W., Ellis, K.P., Tovey, C.A., Xu, H., Automated Process Planning for Printed Circuit Card Assembly, *IIE Transactions*, **24(4)**, 18–30, 1992.

[94] Mehrotra, A., Trick, M.A., A Column Generation Approach for Graph Coloring, *INFORMS Journal on Computing*, **8**, 344–354, 1997.

[95] Montgomery, D.C., editor, *Design and Analysis of Experiments*, 6th edition, John Wiley and Sons, New York, 2005.

[96] Nance, R., Moose, R., Foutz, R., A Statistical Technique for Comparing Heuristics: An Example from Capacity Assignment Strategies in Computer Network Design, *Communications of the ACM*, **30(5)**, 430–442, 1987.

[97] Pan, J.C., Wu, C., Single Machine Group Scheduling to Minimize Mean Flow Time Subject to due Date Constraints, *Production Planning and Control*, **9(4)**, 336–370, 1998.

[98] Ponnambalam, S.G., Aravindan, P., Reddy, R.R., Anaylsis of Group Scheduling Heuristics in a Manufacturing Cell, *International Journal of Advanced Manufacturing Technology*, **15(12)**, 914–932, 1999.

[99] Proust, C., Gupta, J.N.D., Deschamps, V., Flowshop Scheduling with Set–up, Processing and Removal Time Separated, *International Journal of Production Research*, **29**, 479–493, 1991.

[100] Radharamanan, R., A Heuristic Algorithm for Group Scheduling, *Computers and Industrial Engineering*, **11(2–3)**, 204–208, 1986.

[101] Rinnooy Kan, A.H.G., editor, *Machine Scheduling Problems*, Martinus Nijhoff, The Hauge, 1976.

[102] Rossetti, M.D., Stanford, K.J.A., Group Sequencing a PCB Assembly System via an Expected Sequence Dependent Setup Heuristic, *Computers and Industrial Engineering*, **45(1)**, 231–254, 2003.

[103] Sadiq, M., Landers, T.L., Taylor, G.D., 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, 1993.

[104] SAS Institute Inc., *SAS : The SAS System for Windows, Release 8.02*, Cary, NC, USA, 2004.

[105] Savelsbergh, M., A Branch–and–Price Algorithm for the Generalized Assignment Problem, *Operations Research*, **45(6)**, 831–841, 1997.

[106] Schaller, J.E., A New Lower Bound for the Flow Shop Group Scheduling Problem, *Computers and Industrial Engineering*, **41**, 151–161, 2001.

[107] Schaller, J.E., Gupta, J.N.D., Vakharia, A.J., Scheduling a Flow Line Manufacturing Cell with Sequence–Dependent Family Setup Times, *European Journal of Operational Research*, **125(2)**, 324–339, 2000.

[108] Schutten, J.M.J., Van de Velde, S.L., Zijm, W.H.M., Single–Machine Scheduling with Release Dates, Due Dates and Family Setup Times, *Management Science*, **42(8)**, 1165–1174, 1996.

[109] Smed, J., Johnsson, M., Puranen, M., Leipälä, T., Nevalainen, O., Job Grouping in Surface Mounted Component Printing, *Robotics and Computer–Integrated Manufacturing*, **15(1)**, 39–49, 1999.

[110] Smed, J., Johtela, T., Puranen, M., Nevalainen, O., An Interactive System for Schduling Jobs in Electronics Assembly, *International Journal of Advanced Manufacturing Technology*, **16**, 450–459, 2000.

[111] Soumis, F., Decomposition and Column Generation, in Dell'Amico, M., Maffioli, F., Martello, S., editors, *Annotated Bibliography in Combinatorial Optimization*, John Wiley & Sons, Chichester, 115–126, 1997.

[112] Statistical Graphics Corporation, *STATGRAPHICS PLUS*, version 5.1, Rockville, MD, USA, 2001.

[113] Strusevic, V.A., Group Technology Approach to the Open Shop Scheduling Problem with Batch Setup Times, *Operations Research Letters*, **26(4)**, 181–192, 2000.

[114] Sule, D.R., Sequencing n Jobs on Two Machines with Setup, Processing and Removal Times Separated, *Naval Research Logistics Quarterly*, **29**, 517–519, 1982.

[115] Tang, C.S., Denardo, E.V., Models Arising from a Flexible Manufacturing Machine, Part I: Minimization of the Number of Tool Switches, *Operations Research*, **36(5)**, 767–777, 1988.

[116] Teng, S.-H.G, Garimella, S.S., Manufacturing Cost Modeling in Printed Wiring Board Assembly, *Journal of Manufacturing Systems*, **17(2)**, 87–96, 1998.

[117] Uzsoy R., Lee C.Y., Martin–Vega L.A., A Review of Production Planning and Scheduling Models in the Semiconductor Industry Part I: System Characteristics, Performance Evaluation and Production Planning, *IIE Transactions*, **26(5)**, 47–61, 1992.

[118] Uzsoy, R., Lee, C.Y., Martin–Vega, L.A., A Review of Production Planning and Scheduling Models in the Semiconductor Industry Part II: Shop–Floor Control, *IIE Transactions*, **26(5)**, 44–55, 1994.

[119] Wagner, H.M., An Integer Linear–Programming Model for Machine Scheduling, *Naval Research Logistics Quarterly*, **6(2)**, 131–140, 1959.

[120] Wilhelm, W.E., A Technical Review of Column Generation in Integer Programming, *Optimization and Engineering*, **2**, 159–200, 2001.

[121] Yang, D., Chern, M., Two–Machine Flowshop Group Scheduling Problem, *Computers and Operations Research*, **27**, 975–985, 2000.

[122] Yoshida, T., Hitomi, K., Optimal Two–Stage Production Scheduling with Setup Times Separated, *AIIE Transactions*, **11**, 261–263, 1979.

The following references will be sorted alphabetically...

[123] Vance, P., Barnhart, C., Johnson, E., Newhauser, G., Airline Crew Scheduling: A New Formulation and Decomposition Algorithm, *Operations Research*, **45(2)**, 188–200, 1997.

[124] Van den Akker, J., Hoogeven, J., Van de Velde, S., Parallel Machine Scheduling by Column Generation, *Operations Research*, **47(6)**, 862–872, 1999.

[125] Van den Akker, J., Hoogeven, J., Van de Velde, S., Combining Column Generation and Lagrangean Relaxation to Solve a Single–Machine Common Due Date Problem, *INFORMS Journal on Computing*, **14(1)**, 37–51, 1999.

[126] Van den Akker, J., Hurkens, C., Savelsbergh, M., Time–Indexed Formulations for Machine Scheduling Problems: Column Generation, *INFORMS Journal on Computing*, **12(2)**, 111–124, 2000.

[127] Vanderbeck, F., Computational Study of Column Generation for Bin Packing and Cutting Stock Problems, *Mathematical Programming*, **86**, 565–594, 1999.

[128] Vanderbeck., F., On Dantzig–Wolfe Decomposition in Integer Programming and Ways to Perform Branching in a Branch–and–Price Algorithm, *Operations Research*, **48(1)**, 111–128, 2000.

[129] Vanderbeck., F., Wolsey, L.A., An Exact Algorithm for IP Column Generation, *Operations Research Letters*, **19(4)**, 151–159, 1996.

APPENDICES

APPENDIX A. Johnson's Algorithm

For the 2–machine flowshop scheduling problem, Johnson [69] developed an algorithm that guarantees an optimal makespan. The algorithm is as follows.

Johnson's Algorithm

1. Considering both machines, find the minimum value among the run times of jobs. Break ties arbitrarily. Let the minimum value be the run time of job j on machine m.
2. If $m = 1$ place job j at the first available position in the sequence.
   If $m = 2$ place job j at the last available position in the sequence.
3. Remove job j from further consideration. Stop if there are no more jobs to sequence, else go to step 1.

APPENDIX B. Decomposition for Minimizing the Makespan

This section presents the Dantzig–Wolfe decomposition applied to the multi–machine carryover group–scheduling problem with the objective of minimizing the makespan. The decomposition approach for the problem with the objective of minimizing the total flow time was developed in § 7.2 starting at page 111. The reader is referred to § 7.2 for a detailed description of the formulations given below and the relevant notation.

The integer programming master problem (IMP) with the objective of minimizing the makespan can be presented as follows.

IMP:

$$z^*_{IMP} = \text{Min} \sum_{t=1}^{T_m} C^t_{J,m} \lambda^m_t$$

subject to

$$\sum_{t=1}^{T_k} \left(C^t_{j,k} - \sum_{g=1}^{N} \sum_{b=1}^{n_g} rt_{g,b,k} x^{t,k}_{g,b,j} \right) \lambda^k_t - \sum_{t=1}^{T_{k-1}} C^t_{j,k-1} \lambda^{k-1}_t \geq 0$$

$$k = 2, \ldots, m, \quad j = 1, \ldots, J \qquad \text{(B.1)}$$

$$\sum_{t=1}^{T_k} \lambda^k_t = 1 \qquad k = 1, \ldots, m \qquad \text{(B.2)}$$

$$\lambda^k_t \in \{0, 1\} \qquad k = 1, \ldots, m, \quad t = 1, \ldots, T_k \qquad \text{(B.3)}$$

The objective is to select the one schedule in which the completion time of the last board type on the last machine is minimum. Constraint set (B.1) corresponds to constraint set (4.1.8) in MILP1. It makes sure that the schedules selected on the machines ensure that a board type that is completed on a machine starts operation only after it is completed on the previous machine. Constraint

set (B.2) is the set of convexity constraints and makes sure that we select only one schedule on each machine.

The LP relaxation of IMP with the objective of minimizing the makespan is then as presented below. The variables within parentheses before the constraint sets denote the corresponding dual variables.

LMP:

$$z^*_{LMP} = \text{Min} \sum_{t=1}^{T_m} C^t_{J,m} \lambda^m_t$$

subject to

$$(\alpha_{j,k}) \quad \sum_{t=1}^{T_k} \left(C^t_{j,k} - \sum_{g=1}^{N} \sum_{b=1}^{n_g} rt_{g,b,k} x^{t,k}_{g,b,j} \right) \lambda^k_t - \sum_{t=1}^{T_{k-1}} C^t_{j,k-1} \lambda^{k-1}_t \geq 0$$

$$k = 2, \ldots, m, \quad j = 1, \ldots, J \qquad \text{(B.4)}$$

$$(\delta_k) \quad \sum_{t=1}^{T_k} \lambda^k_t = 1 \qquad k = 1, \ldots, m \qquad \text{(B.5)}$$

$$\lambda^k_t \geq 0 \qquad k = 1, \ldots, m, \quad t = 1, \ldots, T_k \qquad \text{(B.6)}$$

In order to construct the subproblems that are capable of identifying new schedules with the most negative reduced costs on each machine, we present the dual of LMP (DLMP) next.

DLMP:

$$z^*_{DLMP} = \text{Max} \sum_{k=1}^{m} \delta_k$$

subject to

$$\sum_{j=1}^{J} -C^t_{j,1}\alpha_{j,1} + \delta_1 \leq 0 \qquad t = 1, \ldots, T_1 \qquad \text{(B.7)}$$

$$\sum_{j=1}^{J} \left(C^t_{j,k} - \sum_{g=1}^{N} \sum_{b=1}^{n_g} rt_{g,b,k} x^{t,k}_{g,b,j} \right) \alpha_{j,k} - \sum_{j=1}^{J} C^t_{j,k}\alpha_{j,k+1} + \delta_k \leq 0$$

$$k = 2, \ldots, m-1, \quad t = 1, \ldots, T'_k \qquad \text{(B.8)}$$

$$\sum_{j=1}^{J} \left(C^t_{j,m} - \sum_{g=1}^{N} \sum_{b=1}^{n_g} rt_{g,b,m} x^{t,m}_{g,b,j} \right) \alpha_{j,m} + \delta_m \leq C^t_{J,m} \qquad t = 1, \ldots, T_m \qquad \text{(B.9)}$$

$$\alpha_{j,k} \geq 0 \qquad j = 1, \ldots, J, \quad k = 1, \ldots, m \qquad \text{(B.10)}$$

$$\delta_k \text{unrestricted} \qquad k = 1, \ldots, m \qquad \text{(B.11)}$$

Recall that the reduced cost of a variable corresponds to the infeasibility in the associated dual constraint. Also note that constraint set (B.7) is associated only with the master variables defined for machine 1, i.e., δ_1 and λ^t_1 for $t = 1, \ldots, T'_1$, as well as it includes completion time variables defined for machine 1 only. Therefore, we need to solve the following subproblem in order to identify the schedule with the smallest reduced cost on machine 1.

SP(1): $z^*_{SP(1)} = \text{Min} \sum_{j=1}^{J} \alpha_{j,2} C_{j,1} - \delta_1$

subject to

(4.1.2) – (4.1.5)

(4.1.6), (4.1.8) – (4.1.16) for machine 1 only.

Similarly, to identify schedules with the smallest reduced costs on machines 2 through $m-1$, we need to solve SP(k) for $k = 2, \ldots, m-1$, respectively.

$$\textbf{SP}(k): \quad z^*_{SP(k)} = \text{Min} \sum_{j=1}^{J} \left((\alpha_{j,k+1} - \alpha_{j,k}) C_{j,k} + \sum_{g=1}^{N} \sum_{b=1}^{n_g} (\alpha_{j,k} r t_{g,b,k}) x_{g,b,j} \right) - \delta_k$$

subject to

$C_{J,k} \leq \overline{MS}$

(4.1.2) – (4.1.5)

(4.1.6), (4.1.8) – (4.1.16) for machine k only.

Finally, the subproblem on the last machine has the following form.

$$\textbf{SP}(m): \quad z^*_{SP(m)} = \text{Min } (1 - \alpha_{J,m}) C_{J,m} - \sum_{j=1}^{J-1} \alpha_{j,m} C_{j,m} + \sum_{j=1}^{J} \sum_{g=1}^{N} \sum_{b=1}^{n_g} (\alpha_{j,m} r t_{g,b,m}) x_{g,b,j} - \delta_m$$

subject to

$C_{J,m} \leq \overline{MS}$

(4.1.2) – (4.1.5)

(4.1.6), (4.1.8) – (4.1.16) for machine m only.

$\overline{MS}$ is an upper bound on the makespan of the original problem. Such an upper bound is readily available as the makespan of any feasible solution to the original problem. We are using the makespan of the best solution identified by our tabu search algorithms proposed for the original problem.

A stabilization method is also used for the case of minimizing the makespan. Consider introducing $w_{j,k}$ for $j = 1, \ldots, J$ and $k = 1, \ldots, m$ as the artificial variables to constraint set B.1 in the IMP as follows.

IMPA:

$$z^*_{IMPA} = \text{Min} \sum_{t=1}^{T_m} C^t_{J,m}\lambda^m_t + \sum_{j=1}^{J} w_{j,m}$$

subject to

$$(\alpha_{j,k}) \quad \sum_{t=1}^{T_k}\Big(C^t_{j,k} - \sum_{g=1}^{N}\sum_{b=1}^{n_g} rt_{g,b,k}x^{t,k}_{g,b,j}\Big)\lambda^k_t - \sum_{t=1}^{T_{k-1}} C^t_{j,k-1}\lambda^{k-1}_t + w_{j,k} - w_{j,k-1} \geq 0$$

$$k = 2,\ldots,m; \quad j = 1,\ldots,J \tag{B.12}$$

$$(\delta_k) \quad \sum_{t=1}^{T_k}\lambda^k_t = 1 \qquad k = 1,\ldots,m \tag{B.13}$$

$$w_{j,k} \geq 0 \qquad k = 1,\ldots,m; \quad j = 1,\ldots,J \tag{B.14}$$

$$\lambda^k_t \in \{0,1\} \qquad k = 1,\ldots,m; \quad t = 1,\ldots,T_k \tag{B.15}$$

The LP relaxation of the IMPA is obtained as usual by relaxing the integrality restrictions on the master variables:

LMPA:

$$z^*_{LMPA} = \text{Min} \sum_{t=1}^{T_m} C^t_{J,m}\lambda^m_t + \sum_{j=1}^{J} w_{j,m}$$

subject to

$$(\alpha_{j,k}) \quad \sum_{t=1}^{T_k}\Big(C^t_{j,k} - \sum_{g=1}^{N}\sum_{b=1}^{n_g} rt_{g,b,k}x^{t,k}_{g,b,j}\Big)\lambda^k_t - \sum_{t=1}^{T_{k-1}} C^t_{j,k-1}\lambda^{k-1}_t + w_{j,k} - w_{j,k-1} \geq 0$$

$$k = 2,\ldots,m; \quad j = 1,\ldots,J \tag{B.16}$$

$$(\delta_k) \quad \sum_{t=1}^{T_k}\lambda^k_t = 1 \qquad k = 1,\ldots,m \tag{B.17}$$

$$w_{j,k} \geq 0 \qquad k = 1,\ldots,m; \quad j = 1,\ldots,J \tag{B.18}$$

$$\lambda^k_t \geq 0 \qquad k = 1,\ldots,m; \quad t = 1,\ldots,T_k \tag{B.19}$$

Then, the dual of LMPA is given as follows.

DLMPA:

$$z^*_{DLMPA} = \text{Max} \sum_{k=1}^{m} \delta_k$$

subject to

$$\sum_{j=1}^{J} -C^t_{j,1}\alpha_{j,2} + \delta_1 \leq 0 \qquad t = 1, \ldots, T_1 \qquad \text{(B.20)}$$

$$\sum_{j=1}^{J} \left(C^t_{j,k} - \sum_{g=1}^{N} \sum_{b=1}^{n_g} rt_{g,b,k} x^{t,k}_{g,b,j} \right) \alpha_{j,k} - \sum_{j=1}^{J} C^t_{j,k}\alpha_{j,k+1} + \delta_k \leq 0$$

$$k = 2, \ldots, m-1; \quad t = 1, \ldots, T_k \qquad \text{(B.21)}$$

$$\sum_{j=1}^{J} \left(C^t_{j,m} - \sum_{g=1}^{N} \sum_{b=1}^{n_g} rt_{g,b,m} x^{t,m}_{g,b,j} \right) \alpha_{j,m} + \delta_m \leq C^t_{J,m} \quad t = 1, \ldots, T_m \qquad \text{(B.22)}$$

$$\alpha_{j,k+1} - \alpha_{j,k} \geq 0 \qquad j = 1, \ldots, J; \quad k = 2, \ldots, m-1 \qquad \text{(B.23)}$$

$$\alpha_{j,m} \leq 1 \qquad j = 1, \ldots, J \qquad \text{(B.24)}$$

$$\alpha_{j,k} \geq 0 \qquad j = 1, \ldots, J; \quad k = 2, \ldots, m \qquad \text{(B.25)}$$

$$\delta_k \text{ unrestricted} \qquad k = 1, \ldots, m \qquad \text{(B.26)}$$

Note that the first three constraint sets are the same in DLMP and DLMPA but constraint sets B.23 and B.24 in DLMPA, together with B.25, now imply that

$$1 \geq \alpha_{j,m} \geq \alpha_{j,m-1} \geq \cdots \geq \alpha_{j,2} \geq 0 \qquad j = 1, \ldots, J \qquad \text{(B.27)}$$

Remember that SP(1) is always bounded because the coefficients of the completion time variables in its objective function are all nonnegative (conditions (B.6)). Note that, when inequality (B.27) is satisfied, all the objective function coefficients of SP(k), $k = 1, \ldots, m-1$, become nonnegative. However, it is not possible to introduce artificial variables in a form that ensures a bounded solution to SP(m).

For certain values of the dual variable values, solving some of the subproblems may be avoided as in the case for minimizing the total flow time.

Case 1: $\alpha_{j,2} = 0$ for $j = 1, \ldots, J$.

Observe that when all $\alpha_{j,2} = 0$, SP(1) objective function becomes minimizing $-\delta_1$ and the constraint set B.7 in DLMP (B.20 in DLMPA) implies that $\delta_1 \leq 0$. Therefore, SP(1) cannot identify a new column with negative reduced cost. Hence, solving SP(1) can be skipped.

Case 2: $\alpha_{j,k} = 0$ for $j = 1, \ldots, J$ and $k = 2, \ldots, m-1$.

In this case, the objective function of SP(k) for $k = 2, \ldots, m-1$ becomes $-\delta_k$ and the constraint set B.8 in DLMP (B.21 in DLMPA) implies that $\delta_k \leq 0$. Hence, solving SP(k) can be skipped.

Case 3: $\alpha_{j,k+1} = \alpha_{j,k} =$ constant for $j = 1, \ldots, J$ and $k = 2, \ldots, m-1$.

In this case, the objective of SP(k), $k = 2, \ldots, m-1$ becomes, minimizing $\sum_{j=1}^{J} \sum_{g=1}^{N} \sum_{b=1}^{n_g} \alpha_{j,k}(rt_{g,b,m})x_{g,b,j} - \delta_k$. The first term is nothing but the sum of the run times of the board types on machine k multiplied by $\alpha_{j,k}$ (which is the same value for all $j = 1, \ldots, J$). From the constraint set B.8 in DLMP (B.21 in DLMPA), it is clear that $\sum_{j=1}^{J} \sum_{g=1}^{N} \sum_{b=1}^{n_g} \alpha_{j,k} rt_{g,b,k} x_{g,b,j} - \delta_k \geq 0$. Hence, SP($k$) can be skipped since it cannot identify any column with negative reduced cost.

Case 4: $\alpha_{J,m} = 1$ and $\alpha_{j,m} = 0$ for $j = 1, \ldots, J-1$.

SP(m) objective becomes minimizing $\sum_{j=1}^{J} \sum_{g=1}^{N} \sum_{b=1}^{n_g} (rt_{g,b,m})x_{g,b,j} - \delta_m$. The first term is nothing but the sum of the run times of the board types on machine m. From the constraint set B.9 in DLMP (B.22 in DLMPA), it is clear that $\sum_{j=1}^{J} \sum_{g=1}^{N} \sum_{b=1}^{n_g} rt_{g,b,m} x_{g,b,j} - \delta_m \geq 0$. Hence, SP($m$) can be skipped since it cannot identify any column with negative reduced cost.

APPENDIX C. Proofs of Propositions 7.6.1 and 7.6.2

Proof of Proposition 7.6.1

We show that the board types within each board group follow the shortest processing time (SPT) order by considering SP(1) first. As illustrated in Figure C.1, consider two solutions, σ and σ', to SP(1). The two solutions are identical except for the sequence of two board types b and b' of some board group g. Also assume that board type g, b is assigned to slot j and board type g, b' to slot $j+1$ in σ and the other way around in σ'. Since SP(1) is bounded (all the objective function coefficients are nonnegative), the contribution of the completion times of the board types in portions A and B to SP(1) objective function are identical for σ and σ'. Therefore, comparing the contributions of board types g, b and g, b' to the SP(1) objective suffices for proving the proposition for SP(1).

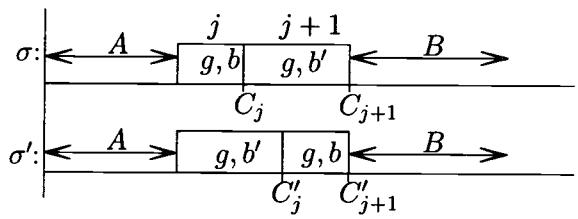

Figure C.1. Two Solutions for SP(1)

Let $z_{SP(1)}(\sigma)$ and $z_{SP(1)}(\sigma')$ be the objective function value of σ in SP(1) and that of σ', respectively. Then,

$$
\begin{aligned}
z_{SP(1)}(\sigma) \le z_{SP(1)}(\sigma') &\Rightarrow \alpha_{j,2}C_j + \alpha_{j+1,2}C_{j+1} \le \alpha_{j,2}C'_j + \alpha_{j+1,2}C'_{j+1} \\
&\Rightarrow \alpha_{j,2}(A + rt_{g,b,1}) + \alpha_{j+1,2}(A + rt_{g,b,1} + rt_{g,b',1}) \\
&\qquad \le \alpha_{j,2}(A + rt_{g,b',1}) + \alpha_{j+1,2}(A + rt_{g,b',1} + rt_{g,b,1}) \\
&\Rightarrow \alpha_{j,2}(A + rt_{g,b,1}) \le \alpha_{j,2}(A + rt_{g,b',1}) \\
&\Rightarrow rt_{g,b,1} \le rt_{g,b',1} \qquad \text{(since } \alpha_{j,2} \ge 0 \qquad j = 1, \ldots, J)
\end{aligned}
$$

$\Rightarrow g, b$ and g, b' are in SPT order

This shows that the board types are in SPT order in an optimal solution, since the above argument holds true for any pair of board types within each group.

A similar approach for the remaining subproblems (i.e., SP(k) for $k = 1, \ldots, m$) can be used to prove the proposition for those subproblems. □

Proof of Proposition 7.6.2

We first note that it is clear that LMPA is a relaxation of LMP (see du Merle et al. [38]).

Here we present a formulation, MILP1A, to the original problem and construct IMPA based on MILP1A. Then we show that, for any sequence, the optimal objective function value of MILP1A is less than or equal to that of MILP1.

Remember from section 7.5 that once we know the sequence of board groups and board types within each group, the optimal objective function value of that sequence can be identified by solving a linear programming problem. Let σ be a sequence of board groups and board types within each group, and $S_{\sigma,k}[g]$ be the amount of carryover sequence–dependent setup time required of board group g in sequence σ on machine k. Also let,

$$\bar{x}_{g,b,j} = \begin{cases} 1, & \text{if board type } b \text{ of group } g \text{ is at slot (position) } j \text{ in sequence } \sigma \\ 0, & \text{otherwise.} \end{cases}$$

and

$$\bar{y}_j = \begin{cases} 1, & \text{if a setup is required at position } j \text{ in sequence } \sigma \\ 0, & \text{otherwise.} \end{cases}$$

The objective function value of σ in MILP1 can be obtained by solving the following LP problem.

TT(MILP1):

$$z^*_{TT(MILP1)} = \text{Min} \sum_{j=1}^{J} C_{j,m} \tag{C.1}$$

subject to

$$C_{j,k} \geq C_{j-1,k} + \sum_{g=1}^{N} S_{\sigma,k}[g] \sum_{b=1}^{n_g} \bar{x}_{g,b,j} \bar{y}_j + \sum_{g=1}^{N} \sum_{b=1}^{n_g} rt_{g,b,k} \bar{x}_{g,b,j} \quad \forall j, \forall k, C_{0,k} = 0 \tag{C.2}$$

$$C_{j,k} \geq C_{j,k-1} + \sum_{g=1}^{N} \sum_{b=1}^{n_g} rt_{g,b,k} \bar{x}_{g,b,j} \qquad \forall j, \forall k, C_{j,0} = 0 \tag{C.3}$$

$$C_{j,k} \geq 0 \qquad \forall j, \forall k$$

Now consider the original formulation MILP1 with the following modification to the constraint set 4.1.7 and to the objective function 4.1.1. In addition to the decision variables introduced in chapter 3, let $w_{j,k}$ (for $j = 1, \ldots, J$ and $k = 1, \ldots, m$) be artificial variables introduced to constraint set 4.1.7 and have objective function coefficients of 1. Let's call this new formulation MILP1A, which can be presented as follows.

MILP1A:

$$z^*_{TT(MILP1A)} = \text{Min} \sum_{j=1}^{J} C_{j,m} + \sum_{j=1}^{J} w_{j,m} \tag{C.4}$$

subject to

$4.1.2 - 4.1.5,\ 4.1.8 - 4.1.16$

$$C_{j,k} \geq C_{j-1,k} + S_{j,k} + \sum_{g=1}^{N} \sum_{b=1}^{n_g} rt_{g,b,k} x_{g,b,j} \qquad \forall j, \forall k, C_{0,k} = 0 \tag{C.5}$$

$$C_{j,k} + w_{j,k} \geq C_{j,k-1} + w_{j,k-1} + \sum_{g=1}^{N} \sum_{b=1}^{n_g} rt_{g,b,k} x_{g,b,j} \qquad \forall j, \forall k, C_{j,0} = 0 \tag{C.6}$$

$$w_{j,k} \geq 0 \qquad \forall j, \forall k$$

Clearly, the master problem constructed by dualizing the constraint set C.6 and relaxing the permutation sequence restriction in MILP1A is equivalent to IMPA; hence

$$z^*_{IMPA} \leq z^*_{MILP1A}. \tag{C.7}$$

Similar to TT(MILP1) above, the optimal objective function value of MILP1A for the sequence σ can be evaluated by the following LP problem.

TT(MILP1A):

$$z^*_{TT(MILP1A)} = \text{Min} \sum_{j=1}^{J} C_{j,m} + \sum_{j=1}^{J} w_{j,m} \tag{C.8}$$

subject to

$$C_{j,k} \geq C_{j-1,k} + \sum_{g=1}^{N} S_{\sigma,k}[g] \sum_{b=1}^{n_g} \bar{x}_{g,b,j}\bar{y}_j + \sum_{g=1}^{N}\sum_{b=1}^{n_g} rt_{g,b,k}\bar{x}_{g,b,j} \quad \forall j, \forall k, C_{0,k}=0 \tag{C.9}$$

$$C_{j,k} + w_{j,k} \geq C_{j,k-1} + w_{j,k-1} + \sum_{g=1}^{N}\sum_{b=1}^{n_g} rt_{g,b,k}\bar{x}_{g,b,j} \qquad \forall j, \forall k, C_{j,0}=0 \tag{C.10}$$

$$C_{j,k} \geq 0 \qquad \forall j, \forall k$$

$$w_{j,k} \geq 0 \qquad \forall j, \forall k$$

Note that the last two terms in the right hand side of constraint set C.2 and the last term in constraint set C.3 are constants. Let's, for simplicity, denote the sum of the second and third terms in the right hand side of constraint set C.2 by $P_{j,k}$ and the last term in the right hand side of constraint set C.3 by $Q_{j,k}$. Let the $\eta_{j,k}$ and $\zeta_{j,k}$ be the dual variables corresponding to constraint set C.2 (C.9) and constraint set C.3 (C.9), respectively. Then, the dual of TT(MILP1) and the dual of TT(MILP1A) can be given as follows.

DTT(MILP1):

$$z^*_{DTT(MILP1)} = \text{Max} \sum_{j=2}^{J}\sum_{k=1}^{m} P_{j,k}\eta_{j,k} + \sum_{j=1}^{J}\sum_{k=2}^{m} Q_{j,k}\zeta_{j,k} \tag{C.11}$$

subject to

$$\eta_{j,k} - \eta_{j+1,k} + \zeta_{j,k} - \zeta_{j,k+1} \leq 0 \qquad j = 1,\ldots,J-1, \quad k = 1,\ldots,m-1 \tag{C.12}$$

$$\eta_{J,k} + \zeta_{J,k} - \zeta_{J,k+1} \leq 0 \qquad k = 1,\ldots,m-1 \tag{C.13}$$

$$\eta_{j,m} - \eta_{j+1,m} + \zeta_{j,m} \leq 1 \qquad j = 1,\ldots,J-1 \tag{C.14}$$

$$\eta_{J,m} + \zeta_{J,m} \leq 1 \qquad k = 1,\ldots,m-1 \tag{C.15}$$

$$\eta_{j,k} \geq 0, \quad \zeta_{j,k} \geq 0 \qquad \forall j, \forall k \tag{C.16}$$

Similarly, the dual of TT(MILP1A) is

DTT(MILP1A):

$$z^*_{DTT(MILP1A)} = \text{Max} \sum_{j=2}^{J}\sum_{k=1}^{m} P_{j,k}\eta_{j,k} + \sum_{j=1}^{J}\sum_{k=2}^{m} Q_{j,k}\zeta_{j,k} \tag{C.17}$$

subject to

$$\eta_{j,k} - \eta_{j+1,k} + \zeta_{j,k} - \zeta_{j,k+1} \leq 0 \qquad j = 1,\ldots,J-1, \quad k = 1,\ldots,m-1 \tag{C.18}$$

$$\eta_{J,k} + \zeta_{J,k} - \zeta_{J,k+1} \leq 0 \qquad k = 1,\ldots,m-1 \tag{C.19}$$

$$\eta_{j,m} - \eta_{j+1,m} + \zeta_{j,m} \leq 1 \qquad j = 1,\ldots,J-1 \tag{C.20}$$

$$\zeta_{j,k} - \zeta_{j,k+1} \leq 0 \qquad j = 1,\ldots,J, \quad k = 1,\ldots,m-1 \tag{C.21}$$

$$\zeta_{j,m} \leq 1 \qquad j = 1,\ldots,J \tag{C.22}$$

$$\eta_{J,m} + \zeta_{J,m} \leq 1 \qquad k = 1,\ldots,m-1 \tag{C.23}$$

$$\eta_{j,k} \geq 0, \quad \zeta_{j,k} \geq 0 \qquad \forall j, \forall k \tag{C.24}$$

From linear programming duality theory, we know that the optimal solution values of primal and dual LP problems are equal. Thus

$$z^*_{TT(MILP1)} = z^*_{DTT(MILP1)} \qquad \text{and} \qquad z^*_{TT(MILP1A)} = z^*_{DTT(MILP1A)}. \tag{C.25}$$

Observe that, DTT(MILP1) and DTT(MILP1A) are identical problems, except for the constraint sets C.21 and C.22 in DTT(MILP1A). Since these constraint sets restrict the problem even further compared to DTT(MILP1),

$$z^*_{DTT(MILP1A)} \leq z^*_{DTT(MILP1)}. \tag{C.26}$$

From C.25 and C.26, we have

$$z^*_{TT(MILP1A)} \leq z^*_{TT(MILP1)}. \tag{C.27}$$

Therefore, for *any* sequence of board groups and board types within each group, the objective function value of MILP1A is less than or equal to that of MILP1, which, combined with C.7, implies

$$z^*_{IMPA} \leq z^*_{MILP1}.$$

□

APPENDIX D. Experimental Results

D.1. Tabu Search versus Actual Optimal Solutions

In this section, experiments are run on two–machine small/medium size problems of type 2. The results are presented in the tables that follow. The description of the column headers in the tables are as follows.

Opt : Optimal solution value identified by CPLEX

TTB : The time the best/optimal solution first identified, in seconds

TT : The total execution time, in seconds

LP : LP–relaxation value of MILP3

LPT : The time to solve LP–relaxation of MILP3, in seconds

N : The number of groups in the problem

IS : Initial solution value identified for the problem

BS : Best solution value identified by the tabu search heuristic

Dev : The percentage deviation of BS from the optimal solution value

The following SAS code is used in the analysis.

```
proc glm;
    class problem ASTH TS;
    model logPD=problem ASTH|TS;
    random problem / test;
    lsmeans TS ASTH /cl adjust=tukey;
    means ASTH TS /cldiff tukey;
run; quit;
```

Table D.1. Optimal Solution Values of MILP3 Formulations by CPLEX

ASTH	Prob.	N	Total Flow Time					Makespan				
			Opt	TTB	TT	LP	LPT	Opt	TTB	TT	LP	LPT
180	S22-1	3	16746	0.06	0.08	8056.2	0.00	5830	0.09	0.09	4918.1	0.00
	S22-2		8514	0.05	0.06	5475.6	0.01	3573	0.06	0.06	3192.0	0.00
	S22-3		17933	0.09	0.09	10079.9	0.02	6236	0.05	0.05	5445.0	0.00
	S22-4	5	47218	1.13	1.17	27073.5	0.03	10465	0.22	0.30	9201.7	0.03
	S22-5		27025	0.58	0.59	19546.4	0.03	6602	1.28	1.29	5947.6	0.03
	S22-6		30972	0.93	0.97	21790.5	0.03	7027	0.70	0.70	6483.2	0.03
	S22-7	6	53867	4.23	7.66	33536.3	0.11	10591	0.92	0.95	9949.1	0.08
	S22-8		39677	8.03	8.52	21816.5	0.19	8188	0.88	0.89	7524.2	0.13
	S22-9		59557	3.78	3.81	40005.3	0.08	12168	0.53	0.56	11855.9	0.13
	S22-10	8	94932	9707.91	10059.40	55007.1	29.20	14621	544.43	544.56	13586.2	24.48
	S22-11		52866	7857.83	8883.72	32861.2	42.66	7848	288.47	289.59	7386.0	26.36
	S22-12		62923	2759.32	2886.31	36299.7	26.56	10972	361.31	361.51	10158.0	22.08
30	S22-1	3	10274	0.08	0.09	4624.3	0.02	3685	0.08	0.08	2818.1	0.00
	S22-2		6608	0.08	0.09	3707.7	0.02	2373	0.05	0.05	2344.0	0.02
	S22-3		12835	0.06	0.08	7594.4	0.02	4586	0.06	0.09	3795.0	0.02
	S22-4	5	32731	0.75	1.16	17668.2	0.05	7631	0.19	0.25	6201.8	0.03
	S22-5		18031	0.53	0.56	11412.6	0.03	4502	0.84	0.92	3971.4	0.03
	S22-6		22887	0.63	0.77	15808.4	0.03	5416	1.17	1.37	5416.0	0.02
	S22-7	6	37754	7.70	8.94	21916.4	0.11	8003	1.31	1.34	7255.4	0.09
	S22-8		26474	4.66	9.27	13539.4	0.08	5788	1.34	1.39	5131.7	0.13
	S22-9		41347	3.48	5.55	27073.9	0.17	8418	0.95	1.22	8186.6	0.09
	S22-10	8	69945	13313.30	15300.60	41468.5	35.27	11414	335.37	336.46	10392.4	25.20
	S22-11		44143	812.30	1711.72	30119.8	46.52	6978	143.94	144.12	6978.0	51.75
	S22-12		40891	3214.14	3221.72	21951.2	36.56	7563	353.79	373.82	6708.0	62.39

Table D.2. Tabu Search versus Optimal with ASTH = 180 (Total flow Time)

Prob.	*N*	IS	TS1				TS2				TS3			
			BS	Dev	TTB	TT	BS	Dev	TTB	Time	BS	Dev	TTB	TT
S22-1	3	16746	16746	0.00	0.00	0.02	16746	0.00	0.00	0.02	16746	0.00	0.00	0.02
S22-2		8514	8514	0.00	0.00	0.02	8514	0.00	0.00	0.02	8514	0.00	0.00	0.00
S22-3		17933	17933	0.00	0.00	0.00	17933	0.00	0.00	0.00	17933	0.00	0.00	0.03
S22-4	5	47218	47218	0.00	0.00	0.02	47218	0.00	0.00	0.19	47218	0.00	0.00	0.03
S22-5		27025	27025	0.00	0.00	0.00	27025	0.00	0.00	0.16	27025	0.00	0.00	0.05
S22-6		31727	30972	0.00	0.02	0.02	30972	0.00	0.02	0.22	30972	0.00	0.00	0.02
S22-7	6	56731	53867	0.00	0.02	0.02	53867	0.00	0.02	0.37	53867	0.00	0.02	0.05
S22-8		39934	39934	0.65	0.00	0.02	39677	0.00	0.11	0.56	39934	0.65	0.00	0.05
S22-9		59581	59557	0.00	0.02	0.02	59557	0.00	0.00	0.72	59557	0.00	0.00	0.13
S22-10	8	98594	96392	1.54	0.03	0.05	94932	0.00	0.28	1.64	96392	1.54	0.05	0.19
S22-11		55852	54338	2.78	0.02	0.03	52866	0.00	0.61	1.25	52866	0.00	0.06	0.14
S22-12		65303	64083	1.84	0.03	0.06	64083	1.84	0.05	1.59	64083	1.84	0.03	0.20

Prob.	*N*	IS	TS4				TS5				TS6			
			BS	Dev	TTB	Time	BS	Dev	TTB	TT	BS	Dev	TTB	TT
S22-1	3	16746	16746	0.00	0.00	0.02	16746	0.00	0.00	0.03	16746	0.00	0.00	0.03
S22-2		8514	8514	0.00	0.00	0.03	8514	0.00	0.00	0.03	8514	0.00	0.00	0.02
S22-3		17933	17933	0.00	0.00	0.02	17933	0.00	0.00	0.02	17933	0.00	0.00	0.02
S22-4	5	47218	47218	0.00	0.00	0.06	47218	0.00	0.00	0.22	47218	0.00	0.00	0.23
S22-5		27025	27025	0.00	0.00	0.05	27025	0.00	0.00	0.30	27025	0.00	0.00	0.16
S22-6		31727	30972	0.00	0.02	0.08	30972	0.00	0.02	0.17	30972	0.00	0.02	0.25
S22-7	6	56731	53867	0.00	0.02	0.22	53867	0.00	0.02	0.56	53867	0.00	0.02	0.38
S22-8		39934	39677	0.00	0.06	0.14	39677	0.00	0.13	0.66	39677	0.00	0.13	0.52
S22-9		59581	59557	0.00	0.00	0.22	59557	0.00	0.00	0.67	59557	0.00	0.02	0.55
S22-10	8	98594	96392	1.54	0.08	0.63	94932	0.00	0.19	1.55	94932	0.00	0.19	1.45
S22-11		55852	53306	0.83	0.28	0.47	52866	0.00	0.55	1.17	52866	0.00	0.84	1.48
S22-12		65303	62923	0.00	0.25	0.47	64083	1.84	0.05	1.36	64083	1.84	0.05	1.86

Table D.3. Tabu Search versus Optimal with ASTH = 30 (Total flow Time)

Prob.	N	IS	TS1				TS2				TS3			
			BS	Dev	TTB	TT	BS	Dev	TTB	Time	BS	Dev	TTB	TT
S22-1	3	10274	10274	0.00	0.00	0.00	10274	0.00	0.00	0.02	10274	0.00	0.00	0.02
S22-2		6792	6608	0.00	0.00	0.00	6608	0.00	0.00	0.02	6608	0.00	0.00	0.00
S22-3		12835	12835	0.00	0.00	0.00	12835	0.00	0.00	0.00	12835	0.00	0.00	0.02
S22-4	5	33751	32731	0.00	0.00	0.02	32731	0.00	0.00	0.27	32731	0.00	0.00	0.03
S22-5		18746	18031	0.00	0.02	0.02	18031	0.00	0.00	0.13	18031	0.00	0.00	0.03
S22-6		23631	22887	0.00	0.00	0.02	22887	0.00	0.02	0.14	22887	0.00	0.02	0.06
S22-7	6	38338	37754	0.00	0.00	0.02	37754	0.00	0.02	0.47	37754	0.00	0.02	0.06
S22-8		26536	26474	0.00	0.02	0.02	26474	0.00	0.02	0.53	26474	0.00	0.02	0.06
S22-9		43167	41347	0.00	0.00	0.02	41347	0.00	0.02	0.58	41347	0.00	0.02	0.05
S22-10	8	70280	69945	0.00	0.02	0.05	69945	0.00	0.02	1.03	69945	0.00	0.03	0.17
S22-11		48648	44617	1.07	0.11	0.12	44143	0.00	0.25	1.20	44617	1.07	0.17	0.33
S22-12		42207	41325	1.06	0.02	0.03	40891	0.00	0.39	1.73	40891	0.00	0.13	0.19

Prob.	N	IS	TS4				TS5				TS6			
			BS	Dev	TTB	Time	BS	Dev	TTB	TT	BS	Dev	TTB	TT
S22-1	3	10274	10274	0.00	0.00	0.02	10274	0.00	0.00	0.03	10274	0.00	0.00	0.05
S22-2		6792	6608	0.00	0.00	0.02	6608	0.00	0.00	0.02	6608	0.00	0.02	0.02
S22-3		12835	12835	0.00	0.00	0.02	12835	0.00	0.00	0.02	12835	0.00	0.00	0.03
S22-4	5	33751	32731	0.00	0.02	0.08	32731	0.00	0.02	0.31	32731	0.00	0.02	0.31
S22-5		18746	18031	0.00	0.02	0.09	18031	0.00	0.00	0.16	18031	0.00	0.00	0.13
S22-6		23631	22887	0.00	0.00	0.06	22887	0.00	0.02	0.16	22887	0.00	0.02	0.17
S22-7	6	38338	37754	0.00	0.02	0.28	37754	0.00	0.02	0.58	37754	0.00	0.00	0.50
S22-8		26536	26474	0.00	0.02	0.14	26474	0.00	0.00	0.50	26474	0.00	0.00	0.64
S22-9		43167	41347	0.00	0.02	0.19	41347	0.00	0.00	0.75	41347	0.00	0.00	0.53
S22-10	8	70280	69945	0.00	0.02	0.44	69945	0.00	0.02	1.38	69945	0.00	0.02	1.30
S22-11		48648	44617	1.07	0.23	0.83	44143	0.00	0.39	1.36	44143	0.00	0.38	1.61
S22-12		42207	40891	0.00	0.44	0.47	40891	0.00	0.38	1.38	40891	0.00	0.47	1.39

Table D.4. Tabu Search versus Optimal with ASTH = 180 (Makespan)

Prob.	N	IS	TS1				TS2				TS3			
			BS	Dev	TTB	TT	BS	Dev	TTB	Time	BS	Dev	TTB	TT
S22-1	3	5935	5830	0.00	0.00	0.02	5830	0.00	0.00	0.00	5830	0.00	0.02	0.02
S22-2		3573	3573	0.00	0.00	0.00	3573	0.00	0.00	0.02	3573	0.00	0.00	0.02
S22-3		6236	6236	0.00	0.00	0.02	6236	0.00	0.00	0.02	6236	0.00	0.00	0.02
S22-4	5	10631	10631	1.59	0.00	0.02	10465	0.00	0.02	0.06	10631	1.59	0.00	0.02
S22-5		6867	6602	0.00	0.00	0.00	6602	0.00	0.02	0.08	6602	0.00	0.00	0.02
S22-6		7289	7109	1.17	0.00	0.02	7027	0.00	0.06	0.13	7109	1.17	0.00	0.03
S22-7	6	11603	10937	3.27	0.02	0.02	10591	0.00	0.16	0.27	10703	1.06	0.02	0.06
S22-8		8368	8264	0.93	0.02	0.03	8188	0.00	0.06	0.13	8264	0.93	0.00	0.05
S22-9		12888	12168	0.00	0.00	0.02	12168	0.00	0.00	0.16	12168	0.00	0.02	0.09
S22-10	8	15434	14689	0.47	0.02	0.05	14689	0.47	0.02	0.33	14689	0.47	0.03	0.14
S22-11		8460	8010	2.06	0.02	0.03	8010	2.06	0.02	0.45	7964	1.48	0.00	0.16
S22-12		11512	11332	3.28	0.02	0.05	11019	0.43	0.09	0.31	11019	0.43	0.02	0.11

Prob.	N	IS	TS4				TS5				TS6			
			BS	Dev	TTB	Time	BS	Dev	TTB	TT	BS	Dev	TTB	TT
S22-1	3	5935	5830	0.00	0.00	0.02	5830	0.00	0.00	0.02	5830	0.00	0.00	0.02
S22-2		3573	3573	0.00	0.00	0.02	3573	0.00	0.00	0.00	3573	0.00	0.00	0.02
S22-3		6236	6236	0.00	0.00	0.03	6236	0.00	0.00	0.02	6236	0.00	0.00	0.02
S22-4	5	10631	10631	1.59	0.00	0.03	10465	0.00	0.02	0.06	10465	0.00	0.02	0.05
S22-5		6867	6602	0.00	0.02	0.08	6602	0.00	0.02	0.06	6602	0.00	0.02	0.06
S22-6		7289	7109	1.17	0.02	0.08	7027	0.00	0.03	0.08	7027	0.00	0.03	0.09
S22-7	6	11603	10591	0.00	0.05	0.09	10591	0.00	0.09	0.16	10591	0.00	0.17	0.27
S22-8		8368	8188	0.00	0.06	0.09	8188	0.00	0.09	0.16	8188	0.00	0.06	0.13
S22-9		12888	12168	0.00	0.02	0.11	12168	0.00	0.02	0.14	12168	0.00	0.02	0.28
S22-10	8	15434	14689	0.47	0.03	0.16	14689	0.47	0.02	0.56	14689	0.47	0.02	0.48
S22-11		8460	7848	0.00	0.19	0.22	8010	2.06	0.02	0.52	8010	2.06	0.02	0.42
S22-12		11512	11019	0.43	0.09	0.13	11019	0.43	0.11	0.41	11019	0.43	0.11	0.45

Table D.5. Tabu Search versus Optimal with ASTH = 30 (Makespan)

Prob.	N	IS	TS1				TS2				TS3			
			BS	Dev	TTB	TT	BS	Dev	TTB	Time	BS	Dev	TTB	TT
S22-1	3	3685	3685	0.00	0.00	0.02	3685	0.00	0.00	0.00	3685	0.00	0.00	0.02
S22-2		2373	2373	0.00	0.00	0.00	2373	0.00	0.00	0.02	2373	0.00	0.00	0.02
S22-3		4586	4586	0.00	0.00	0.02	4586	0.00	0.00	0.00	4586	0.00	0.00	0.02
S22-4	5	7631	7631	0.00	0.00	0.02	7631	0.00	0.00	0.11	7631	0.00	0.00	0.03
S22-5		4586	4527	0.56	0.02	0.02	4502	0.00	0.03	0.08	4527	0.56	0.00	0.02
S22-6		5632	5632	3.99	0.00	0.02	5416	0.00	0.02	0.06	5515	1.83	0.00	0.03
S22-7	6	8153	8123	1.50	0.02	0.03	8063	0.75	0.08	0.28	8123	1.50	0.02	0.05
S22-8		5818	5818	0.52	0.00	0.02	5818	0.52	0.00	0.13	5818	0.52	0.00	0.05
S22-9		8538	8418	0.00	0.00	0.03	8418	0.00	0.02	0.23	8418	0.00	0.02	0.06
S22-10	8	11534	11504	0.79	0.02	0.06	11474	0.53	0.05	0.28	11504	0.79	0.02	0.16
S22-11		7418	6978	0.00	0.02	0.03	6978	0.00	0.02	0.44	6978	0.00	0.03	0.20
S22-12		7612	7582	0.25	0.02	0.03	7582	0.25	0.02	0.38	7582	0.25	0.02	0.11

Prob.	N	IS	TS4				TS5				TS6			
			BS	Dev	TTB	Time	BS	Dev	TTB	TT	BS	Dev	TTB	TT
S22-1	3	3685	3685	0.00	0.00	0.02	3685	0.00	0.00	0.02	3685	0.00	0.00	0.02
S22-2		2373	2373	0.00	0.00	0.02	2373	0.00	0.00	0.00	2373	0.00	0.00	0.03
S22-3		4586	4586	0.00	0.00	0.00	4586	0.00	0.00	0.02	4586	0.00	0.00	0.03
S22-4	5	7631	7631	0.00	0.00	0.08	7631	0.00	0.00	0.08	7631	0.00	0.00	0.06
S22-5		4586	4502	0.00	0.06	0.09	4502	0.00	0.03	0.09	4502	0.00	0.02	0.08
S22-6		5632	5416	0.00	0.03	0.05	5416	0.00	0.03	0.11	5416	0.00	0.03	0.08
S22-7	6	8153	8003	0.00	0.03	0.05	8063	0.75	0.08	0.30	8063	0.75	0.05	0.16
S22-8		5818	5788	0.00	0.03	0.08	5818	0.52	0.00	0.14	5818	0.52	0.00	0.11
S22-9		8538	8418	0.00	0.02	0.11	8418	0.00	0.00	0.16	8418	0.00	0.02	0.28
S22-10	8	11534	11471	0.50	0.22	0.28	11474	0.53	0.05	0.31	11474	0.53	0.05	0.31
S22-11		7418	6978	0.00	0.02	0.17	6978	0.00	0.03	0.30	6978	0.00	0.02	0.30
S22-12		7612	7582	0.25	0.02	0.14	7582	0.25	0.02	0.36	7582	0.25	0.00	0.34

Figures D.2 and D.3 present the plots of used for model adequacy checking after a log–transformation of the percentage deviation. The normality assumption seems to be satisfied. Although the variances seem to vary a little, this variation is not severe, and the analysis is continued with ANOVA.

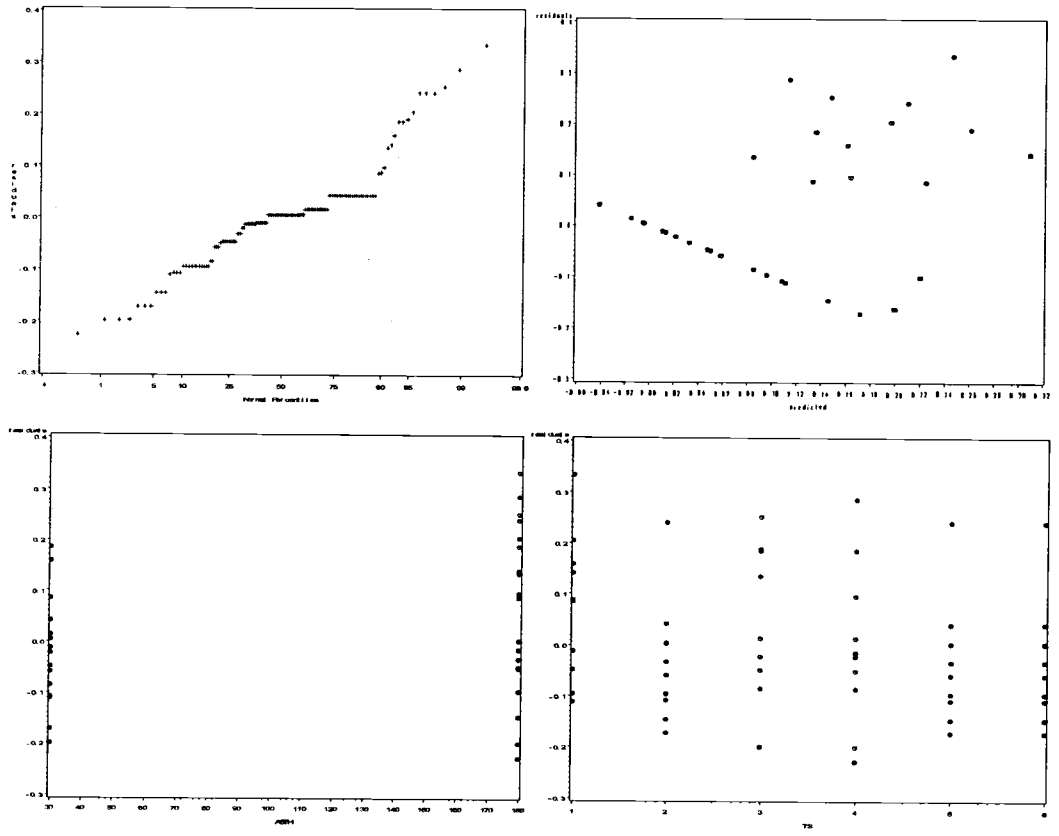

Figure D.2. Model Checking for Log–Transformed Response (Total Flow Time)

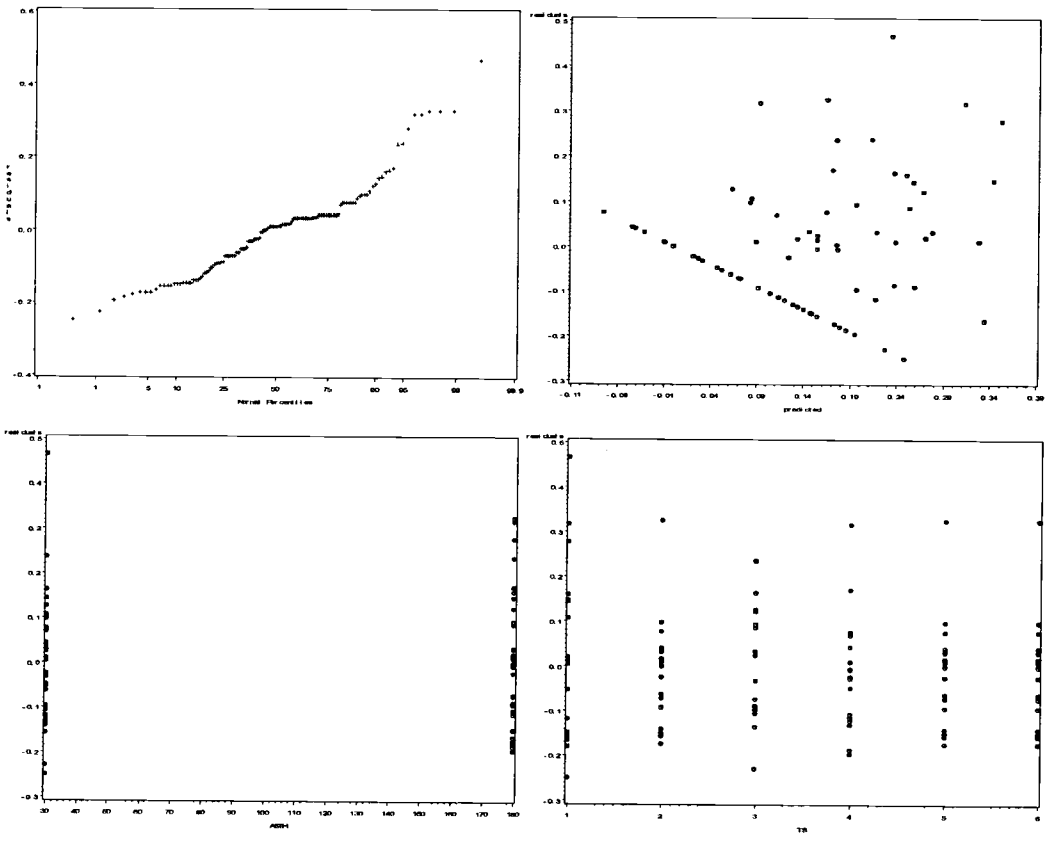

Figure D.3. Model Checking for Log–Transformed Response (Makespan)

D.2. Lower Bounds versus Actual Optimal Solutions

In this section, experiments are run on two–machine small/medium size problems of type 2. The results are presented in the tables that follow. The description of the column headers in the tables are as follows.

N	: The number of groups in the problem
Opt.	: Optimal solution value from Table D.1
LB	: The lower bound (dual bound) identified by the B&P algorithm
Nnodes	: Number of nodes in the B&P tree
Ncols	: Total number of columns generated
TT	: The total execution time, in seconds
Dev	: The percentage deviation of LB from the optimal solution value
Minsetup LB	: The lower bound based on procedure Minsetup

The following SAS code is used in the analysis.

```
proc glm;
    CLASS problem ASTH;
    model PD=problem ASTH;
    random problem / test;
    lsmeans ASTH /cl;
    means ASTH /tukey cldiff;
run; quit;
```

Table D.6. Lower Bounds versus Optimal (Total Flow Time)

ASTH	Prob.	N	Opt.	LB	Nnodes	Ncols	TT	Dev.	Minsetup LB
180	S22-1	3	16746	16746	1	9	0.99	0.00	11739
	S22-2		8514	8514	1	12	1.16	0.00	6574
	S22-3		17933	17933	1	11	1.21	0.00	14168
	S22-4	5	47218	47218	1	18	2.57	0.00	42511
	S22-5		27025	26638	1	15	3.88	1.43	24663
	S22-6		30972	30192	1	16	3.42	2.52	27708
	S22-7	6	53867	53867	1	15	5.13	0.00	42864
	S22-8		39677	39517	1	47	8.08	0.40	35411
	S22-9		59557	58825	1	17	7.69	1.23	48462
	S22-10	8	94932	93556	1	71	166.30	1.45	81441
	S22-11		52866	50779	31	145	18000.00	3.95	45284
	S22-12		62923	62662	1	32	49.77	0.41	53799
30	S22-1	3	10274	9996	1	11	1.62	2.71	8439
	S22-2		6608	6453	1	13	1.89	2.36	5468
	S22-3		12835	12775	1	10	1.83	0.47	11640
	S22-4	5	32731	31768	1	24	2.25	2.94	28561
	S22-5		18031	17511	152	491	3960.77	2.88	16074
	S22-6		22887	22285	113	352	4305.40	2.63	21029
	S22-7	6	37754	37430	1	25	3.16	0.86	33770
	S22-8		26474	25981	78	441	4192.23	1.86	20743
	S22-9		41347	40145	1	39	4.85	2.91	32282
	S22-10	8	69945	68516	1	36	8.31	2.04	61657
	S22-11		44143	43686	1	29	37.19	1.04	35462
	S22-12		40891	39737	95	284	6927.57	2.82	35305

Table D.7. Lower Bounds versus Optimal (Makespan)

ASTH	Prob.	N	Opt.	LB	Nnodes	Ncols	TT	Dev.	Minsetup LB
180	S22-1	3	5830	5755	1	14	1.75	1.29	4955
	S22-2		3573	3486	1	16	2.20	2.43	3393
	S22-3		6236	6236	1	12	1.72	0.00	5696
	S22-4	5	10465	10451	1	22	42.12	0.13	9371
	S22-5		6602	6507	1	25	5.18	1.44	5967
	S22-6		7027	6929	1	32	5.84	1.39	5669
	S22-7	6	10591	10523	1	29	71.64	0.64	9083
	S22-8		8188	8188	1	29	89.81	0.00	6948
	S22-9		12168	12111	1	33	164.30	0.47	11068
	S22-10	8	14689	14354	1	39	625.46	1.83	12734
	S22-11		7848	7740	1	40	65.84	1.38	6480
	S22-12		11019	10972	1	68	186.35	0.00	8992
30	S22-1	3	3685	3655	1	11	1.94	0.81	3505
	S22-2		2373	2373	1	12	1.80	0.00	2343
	S22-3		4586	4586	1	12	2.09	0.00	4496
	S22-4	5	7631	7601	1	21	5.23	0.39	7421
	S22-5		4502	4407	1	23	4.12	2.11	4167
	S22-6		5416	5416	1	20	3.98	0.00	5132
	S22-7	6	8003	7973	1	29	29.51	0.37	7733
	S22-8		5788	5788	1	25	8.36	0.00	5548
	S22-9		8418	8361	1	36	30.64	0.68	8118
	S22-10	8	11471	11384	1	31	856.94	0.26	11084
	S22-11		6978	6758	7	69	18000.00	3.15	5430
	S22-12		7582	7522	1	34	260.98	0.54	7192

Figures D.4 and D.5 present the plots used for model adequacy checking.

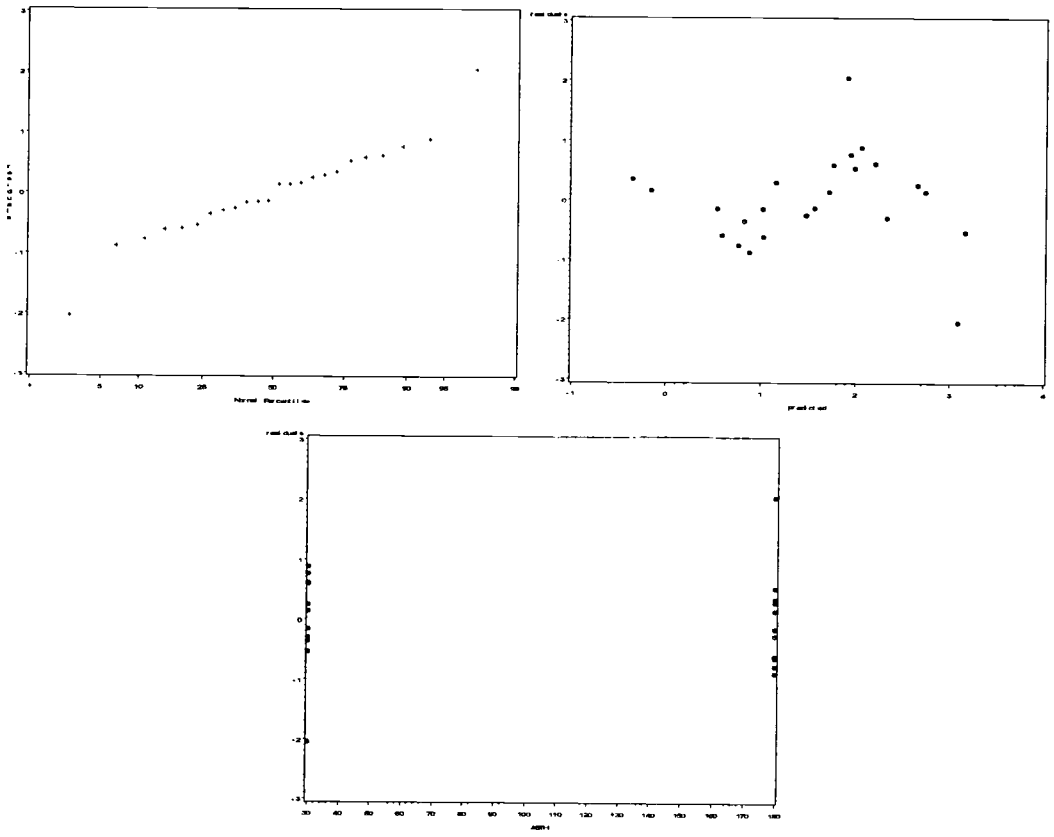

Figure D.4. Model Checking for Percentage Deviation (Total Flow Time)

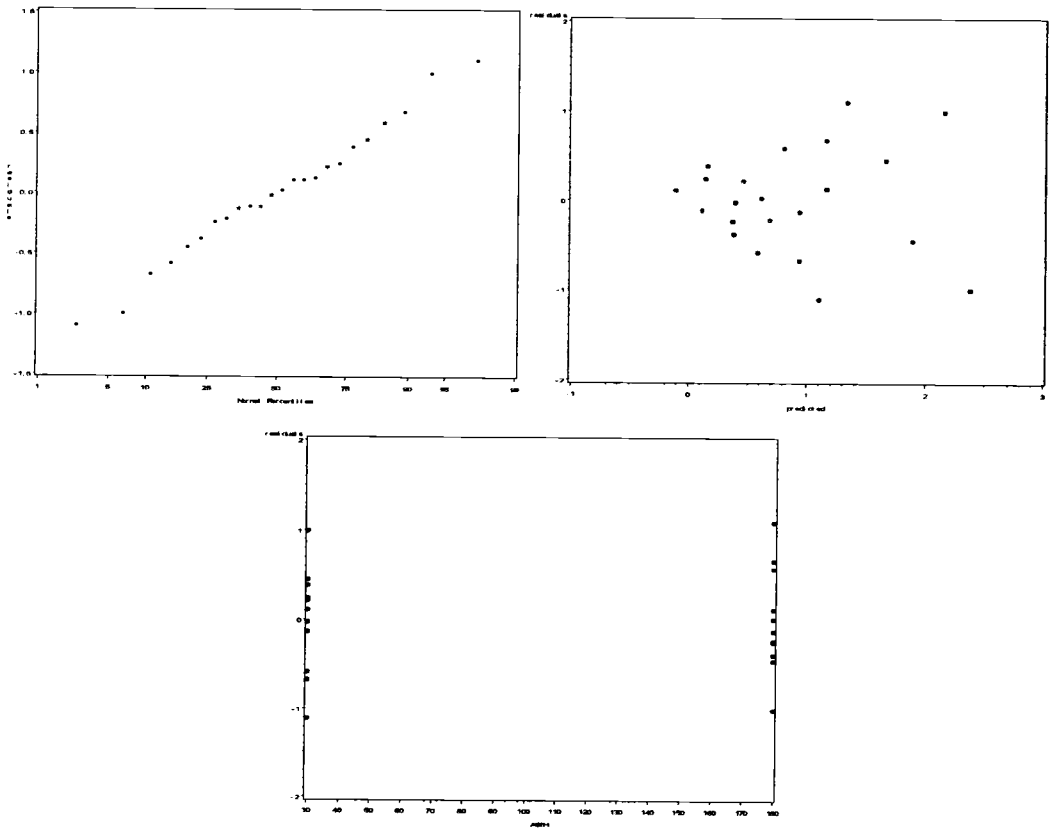

Figure D.5. Model Checking for Percentage Deviation (Makespan)

D.3. Tabu Search versus Proposed Lower Bounds

This section presents the results of the experiments run to test the performance of the various tabu search algorithms with respect to the proposed lower bounds. All the problem instances are solved with the six tabu search algorithms and the B&P algorithm for each of the two measures of performance. The results are presented in the following tables. The description of the column headers in the tables are as follows.

Prob. : The index of the problem

N : The number of groups in the problem

IS : Initial solution value identified for the problem

BS : Best solution value identified by the tabu search heuristic

Dev : The percentage deviation of BS from the lower bound

TTB : The time the best/optimal solution first identified, in seconds

TT : The total execution time, in seconds

ASTH : Average setup time per feeder on HSPM

The following SAS code is used in the analysis.

```
proc mixed;
class PT PS NM D ASTH TS PD;
    model PD = PT|PS|NM|ASTH|TS /ddfm=satterth;
    random D(PT PS NM) ASTH*D(PT PS NM) TS*D(PT PS NM);
    lsmeans ASTH TS /cl pdiff adjust=tukey;
    lsmeans PT*PS*TS /cl pdiff slice = PT*PS adjust = tukey;
run; quit;
```

Table D.8. Small/Medium Size Two–Machine Problems of Type 1 (Total Flow Time)

ASTH = 180															
Prob.	N	IS	TS1				TS2				TS3				
			BS	Dev	TTB	TT	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
S12-1	3	65889	65889	0.00	0.00	0.05	65889	0.00	0.00	0.09	65889	0.00	0.00	2.03	
S12-2	5	197662	196548	0.22	0.05	0.08	196548	0.22	0.17	4.08	196548	0.22	0.03	1.77	
S12-3	6	225000	220861	2.55	0.06	0.31	219284	1.81	0.03	1.94	219284	1.81	0.05	1.50	
S12-4	7	508613	505162	2.84	0.06	0.14	501981	2.19	0.33	16.37	505162	2.84	0.06	1.55	
Prob.	N	IS	TS4				TS5				TS6				
			BS	Dev	TTB	TT	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
S12-1	3	65889	65889	0.00	0.00	2.96	65889	0.00	0.00	0.23	65889	0.00	0.00	0.28	
S12-2	5	197662	196548	0.22	0.06	2.89	196548	0.22	0.17	6.20	196548	0.22	0.17	6.20	
S12-3	6	225000	219284	1.81	0.14	2.28	219284	1.81	0.09	4.84	219284	1.81	0.19	4.69	
S12-4	7	508613	501981	2.19	0.16	2.78	501981	2.19	0.33	16.80	501981	2.19	0.31	16.73	
ASTH = 30															
Prob.	N	IS	TS1				TS2				TS3				
			BS	Dev	TTB	TT	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
S12-1	3	47827	47827	2.23	0.00	0.08	47827	2.23	0.00	0.09	47827	2.23	0.00	0.93	
S12-2	5	141075	141075	3.04	0.03	0.08	140779	2.82	0.03	5.42	141075	3.04	0.02	0.97	
S12-3	6	167567	167567	2.91	0.03	0.09	167265	2.73	0.00	1.36	167567	2.91	0.00	0.81	
S12-4	7	402296	402296	3.42	0.08	0.16	399810	2.78	0.53	12.30	402296	3.42	0.09	1.52	
Prob.	N	IS	TS4				TS5				TS6				
			BS	Dev	TTB	TT	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
S12-1	3	47827	47827	2.23	0.00	1.76	47827	2.23	0.00	0.25	47827	2.23	0.00	0.28	
S12-2	5	141075	140779	2.82	0.00	1.61	140779	2.82	0.03	6.31	140779	2.82	0.03	6.56	
S12-3	6	167567	167265	2.73	0.02	1.63	167265	2.73	0.02	3.47	167265	2.73	0.02	3.28	
S12-4	7	402296	399810	2.78	0.30	3.22	399810	2.78	0.55	17.29	399810	2.78	0.53	18.68	

Table D.9. Small/Medium Size Two–Machine Problems of Type 2 (Total Flow Time)

ASTH = 180															
Prob.	N	IS	TS1				TS2				TS3				
			BS	Dev	TTB	TT	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
S12-1	3	16746	16746	0.00	0.00	0.02	16746	0.00	0.00	0.02	16746	0.00	0.00	0.02	
S12-2	5	47218	47218	0.00	0.00	0.02	47218	0.00	0.00	0.19	47218	0.00	0.00	0.03	
S12-3	6	56731	53867	0.00	0.02	0.02	53867	0.00	0.02	0.37	53867	0.00	0.02	0.05	
S12-4	8	98594	96392	3.03	0.03	0.05	94932	1.47	0.28	1.64	96392	3.03	0.05	0.45	
Prob.	N	IS	TS4				TS5				TS6				
			BS	Dev	TTB	TT	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
S12-1	3	16746	16746	0.00	0.00	0.02	16746	0.00	0.00	0.05	16746	0.00	0.00	0.05	
S12-2	5	47218	47218	0.00	0.00	0.06	47218	0.00	0.00	0.52	47218	0.00	0.00	0.63	
S12-3	6	56731	53867	0.00	0.02	0.32	53867	0.00	0.02	0.86	53867	0.00	0.02	0.88	
S12-4	8	98594	96392	3.03	0.08	0.63	94932	1.47	0.19	1.95	94932	1.47	0.19	1.95	
ASTH = 30															
Prob.	N	IS	TS1				TS2				TS3				
			BS	Dev	TTB	TT	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
S12-1	3	10274	10274	2.78	0.00	0.00	10274	2.78	0.00	0.02	10274	2.78	0.00	0.02	
S12-2	5	33751	32731	3.03	0.00	0.03	32731	3.03	0.00	0.27	32731	3.03	0.00	0.13	
S12-3	6	38338	37754	0.87	0.00	0.03	37754	0.87	0.02	0.47	37754	0.87	0.02	0.26	
S12-4	8	70280	69945	2.09	0.02	0.08	69945	2.09	0.02	1.03	69945	2.09	0.03	0.17	
Prob.	N	IS	TS4				TS5				TS6				
			BS	Dev	TTB	TT	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
S12-1	3	10274	10274	2.78	0.00	0.12	10274	2.78	0.00	0.09	10274	2.78	0.00	0.09	
S12-2	5	33751	32731	3.03	0.02	0.18	32731	3.03	0.02	0.51	32731	3.03	0.02	0.52	
S12-3	6	38338	37754	0.87	0.02	0.28	37754	0.87	0.02	0.88	37754	0.87	0.00	0.95	
S12-4	8	70280	69945	2.09	0.02	0.44	69945	2.09	0.02	1.58	69945	2.09	0.02	1.64	

Table D.10. Large Size Two–Machine Problems of Type 1 (Total Flow Time)

ASTH = 180															
Prob.	N	IS	TS1				TS2				TS3				
			BS	Dev	TTB	TT	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
L12-1	9	674590	664116	8.2	0.20	0.38	658902	7.36	2.52	27.8	658902	7.36	0.53	17.2	
L12-2	10	670531	667109	8.4	0.17	0.94	653365	6.16	2.56	10.1	655025	6.43	2.47	8.1	
L12-3	11	884726	879417	12.0	0.28	0.59	861673	9.74	4.17	17.9	875306	11.47	0.78	4.7	
L12-4	12	1099679	1098293	11.5	0.30	0.72	1090113	10.64	19.48	25.9	1090113	10.64	0.94	5.5	
Prob.	N	IS	TS4				TS5				TS6				
			BS	Dev	TTB	TT	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
L12-1	9	674590	658902	7.4	1.86	20.96	658902	7.36	2.55	54.1	658902	7.36	2.53	53.9	
L12-2	10	670531	653365	6.2	35.39	12.51	653365	6.16	3.05	28.4	653365	6.16	3.16	30.6	
L12-3	11	884726	861673	9.7	3.22	21.36	861673	9.74	4.20	19.2	861673	9.74	4.14	22.1	
L12-4	12	1099679	1090113	10.6	4.71	32.18	1090113	10.64	19.55	59.5	1090113	10.64	20.80	60.6	
ASTH = 30															
Prob.	N	IS	TS1				TS2				TS3				
			BS	Dev	TTB	TT	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
L12-1	9	524578	521946	5.0	0.08	0.55	519651	4.55	1.23	27.8	519651	4.55	0.36	2.2	
L12-2	10	498602	497119	2.5	0.09	0.57	496105	2.32	0.13	10.8	496105	2.32	0.28	2.2	
L12-3	11	643960	642974	9.2	0.14	0.42	641477	8.96	1.72	22.1	642974	9.22	0.49	4.1	
L12-4	12	809598	806670	9.9	0.09	0.79	806670	9.89	2.38	31.5	806670	9.89	0.34	5.5	
Prob.	N	IS	TS4				TS5				TS6				
			BS	Dev	TTB	TT	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
L12-1	9	524578	519651	4.6	0.80	19.74	519651	4.55	1.25	60.3	519651	4.55	1.23	59.9	
L12-2	10	498602	495593	2.2	0.72	22.00	495593	2.22	0.33	24.8	495593	2.22	0.86	25.3	
L12-3	11	643960	641477	8.9	1.28	39.04	641477	8.96	1.73	51.3	641477	8.96	1.69	51.6	
L12-4	12	809598	806400	9.9	61.74	66.83	806670	9.89	2.42	80.6	806670	9.89	2.36	83.3	

Table D.11. Large Size Two-Machine Problems of Type 2 (Total Flow Time)

ASTH = 180															
Prob.	N	IS	TS1				TS2				TS3				
			BS	Dev	TTB	TT	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
L22-1	10	146500	141558	5.29	0.05	0.11	140252	4.32	0.73	3.55	141558	5.29	0.05	1.22	
L22-2	11	137581	137166	8.05	0.11	0.17	137000	7.92	1.25	3.31	134049	5.60	0.30	0.42	
L22-3	13	207398	196344	10.72	0.17	0.27	192560	8.59	2.39	6.64	195962	10.51	0.69	1.20	
L22-4	14	200383	196611	8.80	0.25	0.38	194382	7.57	0.92	7.14	196611	8.80	0.58	0.89	
Prob.	N	IS	TS4				TS5				TS6				
			BS	Dev	TTB	TT	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
L22-1	10	146500	140252	4.32	0.98	1.06	140252	4.32	0.73	3.90	140252	4.32	0.73	3.83	
L22-2	11	137581	137000	7.92	0.20	1.63	137000	7.92	1.30	4.42	137000	7.92	1.31	4.42	
L22-3	13	207398	192560	8.59	0.39	3.09	192560	8.59	2.41	8.22	192560	8.59	2.41	8.23	
L22-4	14	200383	194611	7.69	0.64	3.73	194382	7.57	0.86	9.19	194382	7.57	0.88	9.20	
ASTH = 30															
Prob.	N	IS	TS1				TS2				TS3				
			BS	Dev	TTB	TT	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
L22-1	10	101909	98696	6.13	0.03	0.08	96530	3.80	0.25	2.41	98696	6.13	0.06	1.27	
L22-2	11	93750	93116	2.40	0.03	0.08	92316	1.52	0.08	3.38	93116	2.40	0.03	0.42	
L22-3	13	133194	131967	5.56	0.03	0.11	129558	3.63	1.84	6.09	129558	3.63	0.27	1.53	
L22-4	14	151134	151032	6.69	0.16	0.27	147135	3.94	6.95	6.95	148776	5.10	0.22	0.72	
Prob.	N	IS	TS4				TS5				TS6				
			BS	Dev	TTB	TT	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
L22-1	10	101909	96624	3.90	0.44	0.89	96530	3.80	0.25	2.97	96530	3.80	0.30	2.92	
L22-2	11	93750	92316	1.52	0.06	1.23	92316	1.52	0.03	4.27	92316	1.52	0.05	4.75	
L22-3	13	133194	129558	3.63	1.67	3.66	129558	3.63	1.92	9.28	129558	3.63	1.91	9.13	
L22-4	14	151134	147135	3.94	0.42	3.88	147135	3.94	7.23	9.25	147135	3.94	7.27	9.28	

Table D.12. Small/Medium Size Three–Machine Problems of Type 1 (Total Flow Time)

ASTH = 180														
Prob.	N	IS	TS1				TS2				TS3			
			BS	Dev	TTB	TT	BS	Dev	TTB	Time	BS	Dev	TTB	TT
S13-1	3	96210	95559	0.94	0.05	0.08	95559	0.94	0.00	0.02	95559	0.94	0.02	0.23
S13-2	5	226119	224536	0.61	0.05	0.09	223723	0.24	0.28	7.91	224536	0.61	0.28	2.31
S13-3	6	280112	280112	6.29	0.05	0.09	279228	5.96	0.03	1.77	280112	6.29	0.13	6.64
S13-4	8	652299	643814	10.57	0.19	0.31	630908	8.36	11.64	16.73	633787	8.85	1.38	4.53
Prob.	N	IS	TS4				TS5				TS6			
			BS	Dev	TTB	Time	BS	Dev	TTB	TT	BS	Dev	TTB	TT
S13-1	3	96210	95559	0.94	0.02	0.11	95559	0.94	0.00	0.13	95559	0.94	0.05	0.44
S13-2	5	226119	223723	0.24	0.09	1.50	223723	0.24	0.28	10.30	223723	0.24	0.28	11.33
S13-3	6	280112	279228	5.96	0.09	3.34	279228	5.96	0.06	3.56	279228	5.96	0.13	3.03
S13-4	8	652299	630908	8.36	13.88	15.38	630908	8.36	12.23	19.33	630908	8.36	12.66	20.41
ASTH = 30														
Prob.	N	IS	TS1				TS2				TS3			
			BS	Dev	TTB	TT	BS	Dev	TTB	Time	BS	Dev	TTB	TT
S13-1	3	79397	78725	3.64	0.00	0.11	78725	3.64	0.00	0.02	78725	3.64	0.00	1.03
S13-2	5	175398	174955	4.20	0.03	0.16	173318	3.22	0.05	7.63	173318	3.22	0.06	1.08
S13-3	6	214630	213829	3.12	0.03	0.38	212316	2.39	0.05	4.22	213829	3.12	0.08	1.77
S13-4	8	498228	495896	1.01	0.11	0.35	497417	1.32	2.14	35.39	495896	1.01	0.30	3.53
Prob.	N	IS	TS4				TS5				TS6			
			BS	Dev	TTB	Time	BS	Dev	TTB	TT	BS	Dev	TTB	TT
S13-1	3	79397	77265	1.72	0.02	0.11	78725	3.64	0.02	0.14	77265	1.72	0.15	0.23
S13-2	5	175398	173318	3.22	0.02	1.56	173318	3.22	0.05	9.94	173318	3.22	0.05	9.80
S13-3	6	214630	211829	2.16	0.20	2.44	212316	2.39	0.14	5.92	212316	2.39	0.25	6.20
S13-4	8	498228	495417	0.91	1.06	16.38	497417	1.32	2.16	66.19	497417	1.32	2.14	72.30

Table D.13. Small/Medium Size Three–Machine Problems of Type 2 (Total Flow Time)

ASTH = 180															
Prob.	N	IS	TS1				TS2				TS3				
			BS	Dev	TTB	TT	BS	Dev	TTB	Time	BS	Dev	TTB	TT	
S23-1	4	35600	35379	1.24	0.05	0.06	35379	1.24	0.03	0.11	35379	1.24	0.05	0.02	
S23-2	5	26266	26266	2.44	0.03	0.05	25946	1.19	0.03	0.16	25946	1.19	0.03	0.23	
S23-3	6	48344	47762	2.45	0.03	0.06	46649	0.07	0.03	0.50	46649	0.07	0.03	0.15	
S23-4	8	113811	113791	0.39	0.05	0.08	113791	0.39	0.02	1.03	113791	0.39	0.05	0.25	
Prob.	N	IS	TS4				TS5				TS6				
			BS	Dev	TTB	Time	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
S23-1	4	35600	35379	1.24	0.03	0.03	35379	1.24	0.03	0.38	35379	1.24	0.02	0.39	
S23-2	5	26098	25946	1.19	0.02	0.25	25946	1.19	0.02	0.30	25946	1.19	0.02	0.29	
S23-3	6	47344	46649	0.07	0.03	0.26	46649	0.07	0.03	0.99	46649	0.07	0.03	1.15	
S23-4	8	113811	113791	0.39	0.02	0.38	113791	0.39	0.02	1.90	113791	0.39	0.02	1.98	
ASTH = 30															
Prob.	N	IS	TS1				TS2				TS3				
			BS	Dev	TTB	TT	BS	Dev	TTB	Time	BS	Dev	TTB	TT	
S23-1	4	27350	27350	1.32	0.00	0.05	27350	1.32	0.00	0.13	27350	1.32	0.00	0.02	
S23-2	5	19280	19280	2.44	0.02	0.06	19209	2.07	0.02	0.33	19209	2.07	0.02	0.63	
S23-3	6	35392	35392	2.44	0.00	0.05	35392	2.44	0.00	0.29	35392	2.44	0.00	0.05	
S23-4	8	89799	89799	1.40	0.00	0.08	89527	1.09	0.27	1.11	89799	1.40	0.00	0.16	
Prob.	N	IS	TS4				TS5				TS6				
			BS	Dev	TTB	Time	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
S23-1	4	27350	27350	1.32	0.00	0.03	27350	1.32	0.00	0.34	27350	1.32	0.00	0.34	
S23-2	5	19280	19209	2.07	0.02	0.08	19209	2.07	0.02	0.91	19209	2.07	0.02	0.91	
S23-3	6	35392	35392	2.44	0.00	0.16	35392	2.44	0.00	0.50	35392	2.44	0.00	0.52	
S23-4	8	89799	89527	1.09	0.44	0.47	89527	1.09	0.28	1.59	89527	1.09	0.28	1.62	

Table D.14. Large Size Three–Machine Problems of Type 1 (Total Flow Time)

ASTH = 180														
Prob.	N	IS	TS1				TS2				TS3			
			BS	Dev	TTB	TT	BS	Dev	TTB	Time	BS	Dev	TTB	TT
L13-1	9	578921	577188	10.5	0.11	0.28	571697	9.41	0.13	13.8	577188	10.46	0.19	5.4
L13-2	10	662458	653959	12.5	0.27	0.52	639631	10.04	1.53	16.3	639631	10.04	2.31	7.9
L13-3	11	324857	301986	12.1	0.11	0.17	296854	10.17	1.61	3.7	299306	11.08	0.22	3.4
L13-4	12	1230356	1209430	12.9	0.50	1.02	1195086	11.52	19.66	38.7	1208430	12.77	8.27	13.7
Prob.	N	IS	TS4				TS5				TS6			
			BS	Dev	TTB	Time	BS	Dev	TTB	TT	BS	Dev	TTB	TT
L13-1	9	578921	571697	9.4	0.64	8.08	571697	9.41	0.16	38.3	571697	9.41	0.20	41.1
L13-2	10	662458	639026	9.9	32.81	25.00	639631	10.04	1.52	34.6	639631	10.04	2.11	32.1
L13-3	11	324857	297320	10.3	1.08	4.19	296854	10.17	1.73	6.9	296854	10.17	1.72	6.9
L13-4	12	1230356	1195086	11.5	9.06	53.64	1195086	11.52	20.28	45.5	1195086	11.52	21.04	47.9
ASTH = 30														
Prob.	N	IS	TS1				TS2				TS3			
			BS	Dev	TTB	TT	BS	Dev	TTB	Time	BS	Dev	TTB	TT
L13-1	9	435693	432282	9.6	0.14	0.34	429282	8.80	0.16	13.4	429282	8.80	0.22	4.7
L13-2	10	495222	493508	4.1	0.16	0.41	490887	3.57	1.16	18.6	493508	4.13	0.27	3.4
L13-3	11	292488	285490	12.8	0.09	0.16	282358	11.52	1.86	5.5	285490	12.76	0.20	3.3
L13-4	12	941612	940316	4.0	0.38	0.94	937065	3.64	0.72	43.2	937065	3.64	0.94	16.6
Prob.	N	IS	TS4				TS5				TS6			
			BS	Dev	TTB	Time	BS	Dev	TTB	TT	BS	Dev	TTB	TT
L13-1	9	435693	429282	8.8	0.39	13.47	429282	8.80	0.38	24.8	429282	8.80	0.48	26.9
L13-2	10	495222	490532	3.5	0.86	27.97	490532	3.50	3.09	41.6	490532	3.50	3.08	42.6
L13-3	11	292488	282490	11.6	0.23	3.88	282358	11.52	1.97	5.7	282358	11.52	1.98	5.9
L13-4	12	941612	937065	3.6	0.56	31.33	937065	3.64	0.70	91.6	937065	3.64	0.89	93.5

Table D.15. Large Size Three–Machine Problems of Type 2 (Total Flow Time)

ASTH = 180																
Prob.	N	IS	TS1				TS2				TS3			
			BS	Dev	TTB	TT	BS	Dev	TTB	Time	BS	Dev	TTB	TT
L23-1	10	153851	150760	2.32	0.08	0.14	148133	0.54	3.56	3.58	150760	2.32	0.03	0.25
L23-2	11	128888	124218	10.63	0.11	0.17	119964	6.84	0.09	4.06	119964	6.84	0.08	1.00
L23-3	12	170788	164161	9.72	0.16	0.25	160819	7.48	5.95	5.95	161621	8.02	0.19	1.55
L23-4	14	230445	222713	11.81	0.33	0.49	220017	10.45	1.95	8.81	220674	10.78	0.77	1.28
Prob.	N	IS	TS4				TS5				TS6			
			BS	Dev	TTB	Time	BS	Dev	TTB	TT	BS	Dev	TTB	TT
L23-1	10	153851	150760	2.32	0.08	1.06	148133	0.54	3.73	4.75	148133	0.54	3.75	4.77
L23-2	11	128888	119964	6.84	0.13	1.22	119964	6.84	0.09	5.30	119964	6.84	0.09	5.34
L23-3	12	170788	161621	8.02	0.36	3.45	160819	7.48	6.25	7.25	160819	7.48	6.25	7.27
L23-4	14	230445	220017	10.45	0.94	7.63	220017	10.45	2.03	11.33	220017	10.45	2.06	11.30
ASTH = 30																
Prob.	N	IS	TS1				TS2				TS3			
			BS	Dev	TTB	TT	BS	Dev	TTB	Time	BS	Dev	TTB	TT
L23-1	10	114171	114016	0.68	0.08	0.14	114016	0.68	0.05	3.34	114016	0.68	0.03	0.23
L23-2	11	116290	106838	11.25	0.13	0.20	104824	9.15	0.72	4.06	104838	9.16	0.11	1.34
L23-3	12	120919	119722	6.07	0.08	0.19	118252	4.77	0.06	5.98	118252	4.77	0.06	1.45
L23-4	14	170689	170689	2.29	0.11	0.24	168899	1.22	1.91	8.81	169200	1.40	0.09	3.75
Prob.	N	IS	TS4				TS5				TS6			
			BS	Dev	TTB	Time	BS	Dev	TTB	TT	BS	Dev	TTB	TT
L23-1	10	114171	114016	0.68	0.06	1.13	114016	0.68	0.05	5.12	114016	0.68	0.05	5.55
L23-2	11	116290	104824	9.15	0.20	1.08	104824	9.15	0.78	5.30	104824	9.15	0.78	5.28
L23-3	12	120919	118252	4.77	0.11	2.05	118252	4.77	0.08	7.33	118252	4.77	0.06	7.28
L23-4	14	170327	168899	1.22	0.19	6.77	168899	1.22	1.98	14.23	168899	1.22	1.98	14.93

Figure D.6 presents the plots of used to check whether the assumptions of ANOVA are satisfied.

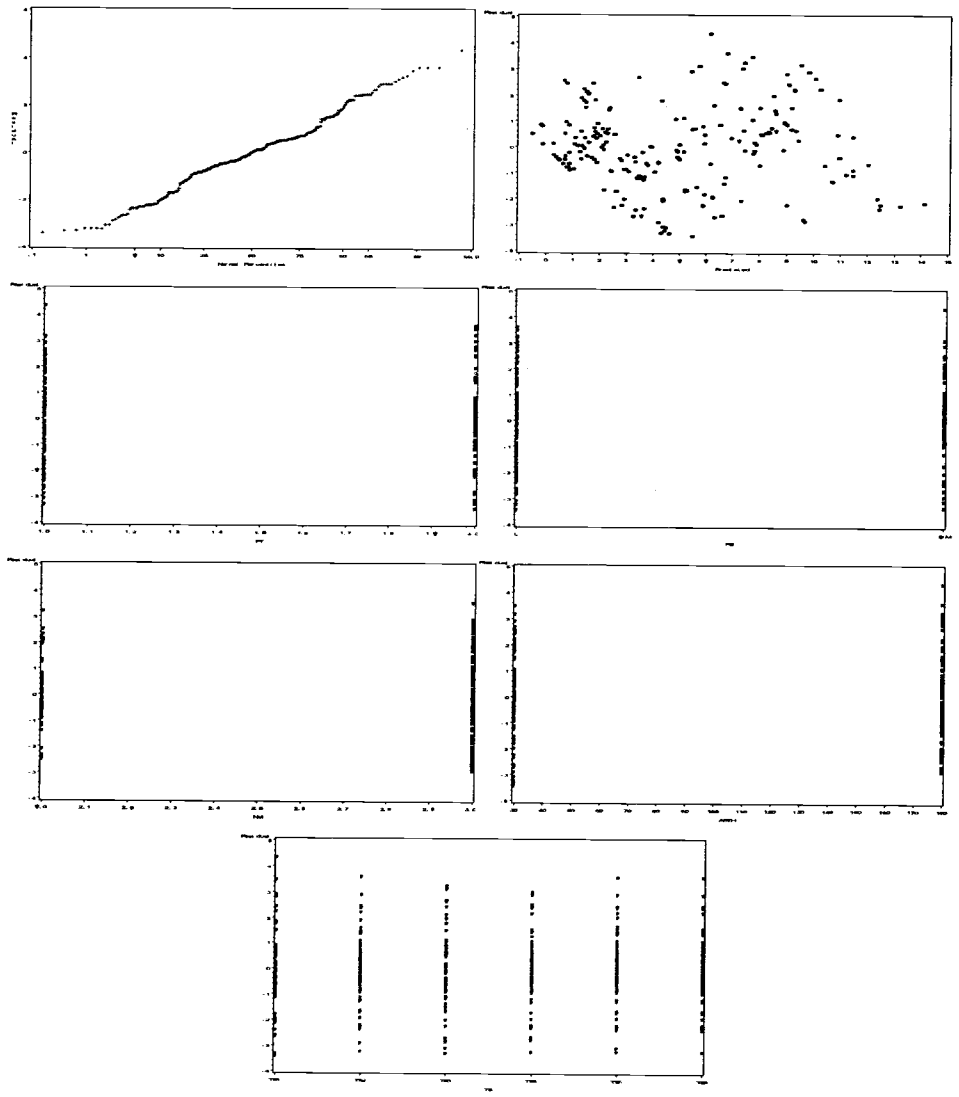

Figure D.6. Model Checking for Tabu Search versus Lower Bounds (Total Flow Time)

Table D.16. Small/Medium Size Two–Machine Problems of Type 1 (Makespan)

ASTH = 180															
Prob.	N	IS	TS1				TS2				TS3				
			BS	Dev	TTB	TT	BS	Dev	TTB	Time	BS	Dev	TTB	TT	
S12-1	3	11254	11074	0.00	0.03	0.06	11074	0.00	0.02	0.95	11074	0.00	0.02	0.31	
S12-2	5	20134	19978	2.00	0.02	0.03	19594	0.04	0.70	2.02	19594	0.04	0.12	0.25	
S12-3	6	19402	19402	2.86	0.06	0.08	19052	1.01	0.08	3.58	19402	2.86	0.14	0.89	
S12-4	7	35439	33895	4.22	0.09	0.14	33674	3.54	1.11	5.93	33895	4.22	0.25	1.70	
Prob.	N	IS	TS4				TS5				TS6				
			BS	Dev	TTB	Time	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
S12-1	3	11254	11074	0.00	0.02	0.71	11074	0.00	0.02	1.78	11074	0.00	0.02	1.93	
S12-2	5	20134	19594	0.04	0.16	0.85	19594	0.04	0.72	3.11	19594	0.04	0.72	3.36	
S12-3	6	19402	19052	1.01	0.16	1.90	19052	1.01	0.23	4.67	19052	1.01	0.27	4.81	
S12-4	7	35439	33535	3.11	0.63	3.01	33674	3.54	1.11	8.33	33674	3.54	1.11	8.84	
ASTH = 30															
Prob.	N	IS	TS1				TS2				TS3				
			BS	Dev	TTB	TT	BS	Dev	TTB	Time	BS	Dev	TTB	TT	
S12-1	3	8554	8524	0.00	0.02	0.10	8524	0.00	0.02	1.05	8524	0.00	0.00	0.93	
S12-2	5	15184	15178	0.61	0.02	0.21	15094	0.05	0.48	2.16	15178	0.61	0.05	0.89	
S12-3	6	14752	14722	1.80	0.03	0.06	14722	1.80	0.05	1.58	14722	1.80	0.06	0.81	
S12-4	7	27489	27309	2.74	0.03	0.09	27129	2.07	4.47	5.55	27309	2.74	0.09	0.98	
Prob.	N	IS	TS4				TS5				TS6				
			BS	Dev	TTB	Time	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
S12-1	3	8554	8524	0.00	0.02	0.05	8524	0.00	0.02	3.08	8524	0.00	0.02	3.08	
S12-2	5	15184	15094	0.05	0.23	0.28	15094	0.05	0.50	3.20	15094	0.05	0.50	3.20	
S12-3	6	14752	14722	1.80	0.09	0.58	14722	1.80	0.13	3.56	14722	1.80	0.14	3.72	
S12-4	7	27489	27189	2.29	0.58	1.06	27129	2.07	4.50	8.64	27129	2.07	4.55	8.99	

Table D.17. Small/Medium Size Two–Machine Problems of Type 2 (Makespan)

ASTH = 180														
Prob.	N	IS	TS1				TS2				TS3			
			BS	Dev	TTB	TT	BS	Dev	TTB	Time	BS	Dev	TTB	TT
S12-1	3	5935	5830	1.30	0.00	0.02	5830	1.30	0.00	0.00	5830	1.30	0.02	0.02
S12-2	5	10631	10631	1.72	0.00	0.02	10465	0.13	0.02	0.06	10631	1.72	0.00	0.02
S12-3	6	11603	10937	3.93	0.02	0.02	10591	0.65	0.16	0.27	10703	1.71	0.02	0.26
S12-4	8	15434	14689	2.33	0.02	0.05	14689	2.33	0.02	0.93	14689	2.33	0.03	0.14
Prob.	N	IS	TS4				TS5				TS6			
			BS	Dev	TTB	Time	BS	Dev	TTB	TT	BS	Dev	TTB	TT
S12-1	3	5935	5830	1.30	0.00	0.22	5830	1.30	0.00	0.02	5830	1.30	0.00	0.02
S12-2	5	10631	10631	1.72	0.00	0.13	10465	0.13	0.02	0.16	10465	0.13	0.02	0.15
S12-3	6	11603	10591	0.65	0.05	0.39	10591	0.65	0.09	0.76	10591	0.65	0.17	0.67
S12-4	8	15434	14689	2.33	0.03	0.26	14689	2.33	0.02	1.56	14689	2.33	0.02	1.48
ASTH = 30														
Prob.	N	IS	TS1				TS2				TS3			
			BS	Dev	TTB	TT	BS	Dev	TTB	Time	BS	Dev	TTB	TT
S12-1	3	3685	3685	0.82	0.00	0.02	3685	0.82	0.00	0.12	3685	0.82	0.00	0.02
S12-2	5	7631	7631	0.39	0.00	0.02	7631	0.39	0.00	0.21	7631	0.39	0.00	0.03
S12-3	6	8153	8123	1.88	0.02	0.03	8063	1.13	0.08	0.28	8123	1.88	0.02	0.05
S12-4	7	11534	11504	1.05	0.02	0.06	11474	0.79	0.05	0.38	11504	1.05	0.02	0.16
Prob.	N	IS	TS4				TS5				TS6			
			BS	Dev	TTB	Time	BS	Dev	TTB	TT	BS	Dev	TTB	TT
S12-1	3	3685	3685	0.82	0.00	0.02	3685	0.82	0.00	0.32	3685	0.82	0.00	0.12
S12-2	5	7631	7631	0.39	0.00	0.08	7631	0.39	0.00	0.58	7631	0.39	0.00	0.73
S12-3	6	8153	8003	0.38	0.03	0.05	8063	1.13	0.08	0.70	8063	1.13	0.05	0.86
S12-4	7	11534	11471	0.76	0.22	0.28	11474	0.79	0.05	0.11	11474	0.79	0.05	0.21

Table D.18. Large Size Two–Machine Problems of Type 1 (Makespan)

ASTH = 180															
Prob.	N	IS	TS1				TS2				TS3				
			BS	Dev	TTB	TT	BS	Dev	TTB	Time	BS	Dev	TTB	TT	
L12-1	9	37581	37123	8.61	0.09	0.22	36361	6.38	0.52	12.26	36361	6.38	0.45	1.89	
L12-2	10	38392	37312	9.99	0.12	0.28	37132	9.46	0.66	4.35	37132	9.46	1.61	2.78	
L12-3	11	43024	41404	10.60	0.31	0.56	40504	8.20	12.25	14.51	40864	9.16	4.64	5.55	
L12-4	12	48185	47109	11.83	0.98	1.34	45728	8.56	15.54	37.52	46025	9.26	3.81	9.25	
Prob.	N	IS	TS4				TS5				TS6				
			BS	Dev	TTB	Time	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
L12-1	9	37581	36361	6.38	0.28	3.08	36361	6.38	0.52	17.41	36361	6.38	0.52	18.34	
L12-2	10	38392	37312	9.99	0.41	5.19	37132	9.46	0.89	16.53	37132	9.46	0.89	17.66	
L12-3	11	43024	40438	8.02	5.42	9.20	40504	8.20	12.50	39.67	40504	8.20	12.40	39.20	
L12-4	12	48185	46025	9.26	3.45	14.60	45728	8.56	15.06	61.20	45728	8.56	15.60	61.70	
ASTH = 30															
Prob.	N	IS	TS1				TS2				TS3				
			BS	Dev	TTB	TT	BS	Dev	TTB	Time	BS	Dev	TTB	TT	
L12-1	9	30381	30351	4.54	0.05	0.17	30231	4.13	11.79	9.36	30351	4.54	0.17	2.77	
L12-2	10	29542	29362	4.82	0.13	0.27	29332	4.72	0.81	5.83	29332	4.72	1.52	2.70	
L12-3	11	32044	31811	6.54	0.50	0.75	31744	6.32	3.21	19.93	31744	6.32	1.80	4.78	
L12-4	12	36485	36125	13.22	1.00	1.36	36125	13.22	26.11	41.02	36125	13.22	3.88	9.25	
Prob.	N	IS	TS4				TS5				TS6				
			BS	Dev	TTB	Time	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
L12-1	9	30381	30321	4.44	1.81	7.20	30231	4.13	11.59	18.89	30231	4.13	11.39	17.59	
L12-2	10	29542	29362	4.82	0.39	6.44	29332	4.72	1.91	12.41	29332	4.72	2.08	13.84	
L12-3	11	32044	31744	6.32	1.61	8.73	31744	6.32	3.30	49.36	31744	6.32	3.40	50.94	
L12-4	12	36485	36098	13.14	11.28	24.41	36125	13.22	25.73	88.05	36125	13.22	25.56	89.80	

Table D.19. Large Size Two–Machine Problems of Type 2 (Makespan)

ASTH = 180															
Prob.	N	IS	TS1				TS2				TS3				
			BS	Dev	TTB	TT	BS	Dev	TTB	Time	BS	Dev	TTB	TT	
L22-1	10	19865	19325	10.51	0.03	0.09	18965	8.45	0.53	1.72	19325	10.51	0.03	0.19	
L22-2	11	19299	17729	9.97	0.13	0.19	17249	7.00	0.25	2.19	17729	9.97	0.11	0.30	
L22-3	13	22057	20437	12.71	0.13	0.22	19717	8.74	0.34	4.02	20077	10.73	0.69	0.80	
L22-4	14	23002	22282	16.86	0.05	0.20	20667	8.39	1.36	5.69	21922	14.97	0.22	0.56	
Prob.	N	IS	TS4				TS5				TS6				
			BS	Dev	TTB	Time	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
L22-1	10	19865	18965	8.45	0.31	0.36	18965	8.45	0.53	3.75	18965	8.45	0.56	3.77	
L22-2	11	19299	17249	7.00	0.31	0.58	17249	7.00	0.27	4.25	17249	7.00	0.27	4.25	
L22-3	13	22057	19717	8.74	0.52	0.78	19717	8.74	0.38	7.13	19717	8.74	0.36	6.33	
L22-4	14	23002	21382	12.14	0.38	0.75	20667	8.39	1.44	9.81	20667	8.39	1.45	9.83	
ASTH = 30															
Prob.	N	IS	TS1				TS2				TS3				
			BS	Dev	TTB	TT	BS	Dev	TTB	Time	BS	Dev	TTB	TT	
L22-1	10	13822	13685	5.68	0.05	0.09	13565	4.76	0.23	1.83	13685	5.68	0.05	0.90	
L22-2	11	13749	13379	4.85	0.09	0.14	13330	4.47	1.31	2.33	13379	4.85	0.12	0.91	
L22-3	13	15307	15216	10.73	0.19	0.25	15007	9.21	0.30	3.19	15007	9.21	0.30	0.94	
L22-4	14	16552	16432	2.69	0.03	0.13	16252	1.56	0.77	4.72	16372	2.31	0.23	0.85	
Prob.	N	IS	TS4				TS5				TS6				
			BS	Dev	TTB	Time	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
L22-1	10	13822	13565	4.76	0.06	1.97	13565	4.76	0.23	1.88	13565	4.76	0.25	1.89	
L22-2	11	13749	13354	4.66	0.63	1.70	13330	4.47	1.42	3.45	13330	4.47	1.42	3.45	
L22-3	13	15307	15007	9.21	0.33	2.47	15007	9.21	0.33	7.31	15007	9.21	0.33	7.40	
L22-4	14	16552	16222	1.37	0.45	2.02	16252	1.56	0.83	9.88	16252	1.56	0.81	9.98	

Table D.20. Small/Medium Size Three–Machine Problems of Type 1 (Makespan)

ASTH = 180															
Prob.	N	IS	TS1				TS2				TS3				
			BS	Dev	TTB	TT	BS	Dev	TTB	Time	BS	Dev	TTB	TT	
S13-1	3	14037	13255	2.46	0.00	0.02	13451	3.97	0.02	1.03	13255	2.46	0.00	0.95	
S13-2	5	20396	19892	1.46	0.02	0.03	19718	0.58	0.91	3.33	19718	0.58	0.03	1.93	
S13-3	6	23253	23073	3.21	0.02	0.06	22533	0.79	0.27	2.12	22533	0.79	0.03	2.33	
S13-4	8	38356	37480	9.35	0.09	0.22	36914	7.70	5.22	11.73	37021	8.01	0.28	2.96	
Prob.	N	IS	TS4				TS5				TS6				
			BS	Dev	TTB	Time	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
S13-1	3	14037	13109	1.33	0.03	2.36	13451	3.97	0.03	2.78	13109	1.33	0.03	2.94	
S13-2	5	20396	19682	0.39	0.25	5.53	19718	0.58	0.89	5.36	19718	0.58	0.91	5.38	
S13-3	6	23253	22476	0.54	0.25	5.53	22533	0.79	0.27	4.64	22533	0.79	0.30	4.75	
S13-4	8	38356	37402	9.13	2.19	7.59	36914	7.70	5.30	18.94	36914	7.70	5.27	18.06	
ASTH = 30															
Prob.	N	IS	TS1				TS2				TS3				
			BS	Dev	TTB	TT	BS	Dev	TTB	Time	BS	Dev	TTB	TT	
S13-1	3	11278	10855	4.36	0.02	0.02	10709	2.96	0.02	4.03	10855	4.36	0.02	0.05	
S13-2	5	16526	16118	1.11	0.06	0.08	16118	1.11	0.31	3.66	16118	1.11	0.14	0.38	
S13-3	6	17703	17673	1.53	0.02	0.05	17583	1.02	0.53	6.77	17583	1.02	0.03	0.34	
S13-4	8	31756	32019	8.25	0.39	0.53	31162	5.36	4.38	11.39	31192	5.46	1.08	3.28	
Prob.	N	IS	TS4				TS5				TS6				
			BS	Dev	TTB	Time	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
S13-1	3	11278	10709	2.96	0.02	0.18	10709	2.96	0.02	11.08	10709	2.96	0.03	11.11	
S13-2	5	16526	16082	0.88	0.17	1.18	16118	1.11	0.30	9.72	16118	1.11	0.31	9.75	
S13-3	6	17703	17526	0.69	0.25	1.08	17583	1.02	0.55	16.19	17583	1.02	0.56	16.30	
S13-4	8	31756	31192	5.46	1.23	9.09	31162	5.36	4.39	31.52	31162	5.36	4.44	31.77	

Table D.21. Small/Medium Size Three–Machine Problems of Type 2 (Makespan)

ASTH = 180															
Prob.	N	IS	TS1				TS2				TS3				
			BS	Dev	TTB	TT	BS	Dev	TTB	Time	BS	Dev	TTB	TT	
S23-1	4	9473	9325	3.31	0.03	0.05	9290	2.92	0.00	0.03	9290	2.92	0.02	0.12	
S23-2	5	6363	6363	2.83	0.00	0.05	6363	2.83	0.00	0.35	6363	2.83	0.00	0.22	
S23-3	6	9959	9717	4.52	0.03	0.06	9357	0.65	0.08	0.20	9717	4.52	0.02	0.06	
S23-4	8	18696	17616	1.43	0.05	0.08	17616	1.43	0.05	0.99	17616	1.43	0.05	0.12	
Prob.	N	IS	TS4				TS5				TS6				
			BS	Dev	TTB	Time	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
S23-1	4	9473	9290	2.92	0.02	0.13	9290	2.92	0.02	0.13	9290	2.92	0.03	0.28	
S23-2	5	6363	6363	2.83	0.00	0.32	6363	2.83	0.00	0.25	6363	2.83	0.00	0.49	
S23-3	6	9959	9357	0.65	0.06	0.19	9357	0.65	0.08	0.37	9357	0.65	0.11	0.30	
S23-4	8	18696	17616	1.43	0.05	0.26	17616	1.43	0.03	2.34	17616	1.43	0.08	2.45	
ASTH = 30															
Prob.	N	IS	TS1				TS2				TS3				
			BS	Dev	TTB	TT	BS	Dev	TTB	Time	BS	Dev	TTB	TT	
S23-1	4	7673	6890	1.23	0.03	0.05	6890	1.23	0.00	0.03	6890	1.23	0.02	0.02	
S23-2	5	4782	4782	2.95	0.00	0.05	4782	2.95	0.00	0.05	4782	2.95	0.00	0.13	
S23-3	6	7722	7447	1.28	0.03	0.05	7447	1.28	0.02	0.67	7447	1.28	0.02	0.25	
S23-4	8	14946	14813	1.17	0.06	0.09	14783	0.97	0.09	0.67	14813	1.17	0.03	0.23	
Prob.	N	IS	TS4				TS5				TS6				
			BS	Dev	TTB	Time	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
S23-1	4	7673	6890	1.23	0.02	0.13	6890	1.23	0.02	0.15	6890	1.23	0.02	0.15	
S23-2	5	4782	4782	2.95	0.00	0.13	4782	2.95	0.00	0.16	4782	2.95	0.00	0.26	
S23-3	6	7722	7447	1.28	0.02	0.38	7447	1.28	0.02	1.19	7447	1.28	0.02	1.20	
S23-4	8	14946	14813	1.17	0.05	0.38	14783	0.97	0.09	1.48	14783	0.97	0.09	1.50	

Table D.22. Large Size Three–Machine Problems of Type 1 (Makespan)

ASTH = 180															
Prob.	N	IS	TS1				TS2				TS3				
			BS	Dev	TTB	TT	BS	Dev	TTB	Time	BS	Dev	TTB	TT	
L13-1	9	36011	35761	11.11	0.03	0.17	35221	9.43	1.16	9.56	35581	10.55	0.09	2.20	
L13-2	10	36694	35703	13.03	0.13	0.33	35222	11.51	0.78	11.89	35222	11.51	1.41	6.07	
L13-3	11	23767	23149	13.58	0.03	0.08	22993	12.82	0.33	12.95	23149	13.58	0.03	5.20	
L13-4	12	56996	54476	10.83	0.56	1.00	54476	10.83	1.03	29.02	54476	10.83	1.66	5.97	
Prob.	N	IS	TS4				TS5				TS6				
			BS	Dev	TTB	Time	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
L13-1	9	36011	35221	9.43	1.08	5.89	35221	9.43	1.36	15.17	35221	9.43	1.83	15.42	
L13-2	10	36694	35332	11.86	1.97	16.80	35222	11.51	1.22	21.16	35222	11.51	1.48	21.36	
L13-3	11	23767	23149	13.58	0.03	14.31	22993	12.82	0.34	26.02	22993	12.82	0.36	25.72	
L13-4	12	56996	53748	9.35	8.33	17.64	54476	10.83	1.97	55.53	54275	10.42	19.34	57.89	
ASTH = 30															
Prob.	N	IS	TS1				TS2				TS3				
			BS	Dev	TTB	TT	BS	Dev	TTB	Time	BS	Dev	TTB	TT	
L13-1	9	27971	27841	10.59	0.03	0.17	27721	10.11	1.89	8.44	27751	10.23	2.45	2.81	
L13-2	10	28744	28503	8.60	0.13	0.33	28472	8.48	0.78	13.61	28472	8.48	1.63	4.45	
L13-3	11	30959	29741	10.80	0.08	0.13	29741	10.80	0.13	19.13	29741	10.80	0.11	5.80	
L13-4	12	43166	42588	6.16	0.64	1.08	42588	6.16	1.17	22.39	42588	6.16	1.84	8.98	
Prob.	N	IS	TS4				TS5				TS6				
			BS	Dev	TTB	Time	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
L13-1	9	27971	27781	10.35	0.91	8.91	27721	10.11	2.41	21.86	27721	10.11	3.09	22.05	
L13-2	10	28744	28492	8.55	2.73	12.77	28472	8.48	1.25	30.19	28472	8.48	1.50	31.77	
L13-3	11	30959	29741	10.80	0.13	14.52	29741	10.80	0.13	45.20	29741	10.80	0.13	45.20	
L13-4	12	43166	42545	6.05	2.02	24.06	42588	6.16	1.30	47.70	42545	6.05	17.33	48.06	

Table D.23. Large Size Three–Machine Problems of Type 2 (Makespan)

ASTH = 180															
Prob.	N	IS	TS1				TS2				TS3				
			BS	Dev	TTB	TT	BS	Dev	TTB	Time	BS	Dev	TTB	TT	
L23-1	10	19963	18855	4.95	0.06	0.09	18495	2.94	0.16	2.88	18495	2.94	0.25	0.30	
L23-2	11	16175	16214	9.43	0.00	0.06	15815	6.74	0.02	2.98	15815	6.74	0.02	0.19	
L23-3	12	19337	17897	12.06	0.09	0.16	17537	9.81	1.31	4.95	17897	12.06	0.16	1.48	
L23-4	14	26648	24675	10.48	0.20	0.31	24495	9.68	1.31	6.41	24675	10.48	0.31	2.28	
Prob.	N	IS	TS4				TS5				TS6				
			BS	Dev	TTB	Time	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
L23-1	10	19963	18855	4.95	0.09	0.92	18495	2.94	0.19	5.14	18495	2.94	0.19	5.05	
L23-2	11	16175	15398	3.92	0.11	0.31	15815	6.74	0.02	5.05	15815	6.74	0.03	5.06	
L23-3	12	19337	17537	9.81	0.17	3.95	17537	9.81	1.45	9.11	17537	9.81	1.39	9.98	
L23-4	14	26648	24675	10.48	0.31	6.19	24495	9.68	1.39	8.91	24495	9.68	1.39	8.85	
ASTH = 30															
Prob.	N	IS	TS1				TS2				TS3				
			BS	Dev	TTB	TT	BS	Dev	TTB	Time	BS	Dev	TTB	TT	
L23-1	10	15613	15405	1.72	0.05	0.09	15345	1.32	0.16	1.78	15345	1.32	0.25	0.30	
L23-2	11	15162	13332	9.51	0.08	0.13	13332	9.51	0.13	2.33	13332	9.51	0.11	0.34	
L23-3	12	13787	13547	1.79	0.09	0.16	13517	1.56	0.62	5.73	13547	1.79	0.14	1.47	
L23-4	14	18848	18185	7.88	0.39	0.52	18185	7.88	0.83	6.47	18185	7.88	0.63	3.80	
Prob.	N	IS	TS4				TS5				TS6				
			BS	Dev	TTB	Time	BS	Dev	TTB	TT	BS	Dev	TTB	TT	
L23-1	10	15613	15345	1.32	0.09	0.52	15345	1.32	0.20	2.91	15345	1.32	0.17	2.83	
L23-2	11	15162	13332	9.51	0.13	0.98	13332	9.51	0.17	3.49	13332	9.51	0.13	3.45	
L23-3	12	13787	13517	1.56	0.17	2.89	13517	1.56	0.64	8.81	13517	1.56	0.64	8.11	
L23-4	14	18848	18155	7.70	0.98	4.66	18185	7.88	0.86	10.67	18185	7.88	0.86	10.69	

Figure D.7 presents the plots of used to check whether the assumptions of ANOVA are satisfied.

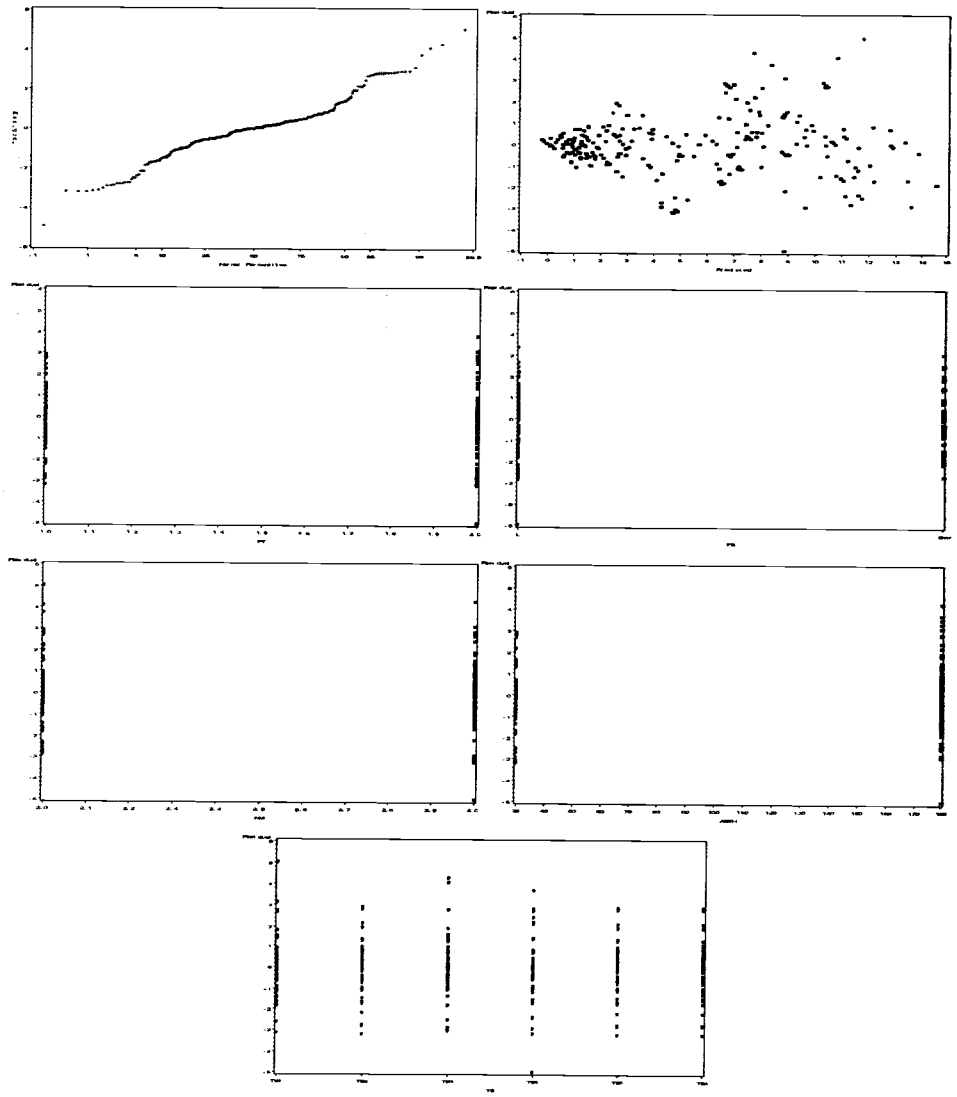

Figure D.7. Model Checking for Tabu Search versus Lower Bounds (Makespan)

The detailed results regarding the B&P algorithms to obtain the lower bounds are presented on the following tables. The description of the column headers in the tables are as follows.

Criteria : Minimizing the Total flow time (TFT) or the Makespan

ASTH : Average setup time per feeder on HSPM

Prob. : The index of the problem

LB : The lower bound obtained from the problem

Nnodes : The number of nodes generated by the B&P algorithm

Ncols : The number of columns generated by the B&P algorithm

TT : The total execution time of the B&P algorithm

Table D.24. Lower Bounds for the Small/Medium Size Two–Machine Problems of Type 1 (Total Flow time)

Criteria	ASTH	Prob.	LB	Nnodes	Ncols	TT
TFT	180	S12-1	65889	1	23	1.45
		S12-2	196113	1	18	42.05
		S12-3	215378	1	35	3985.17
		S12-4	491235	1	48	11615.72
	30	S12-1	46785	1	14	2.66
		S12-2	136912	126	487	18000.00
		S12-3	162827	1	28	1958.20
		S12-4	388986	1	31	57.58
Makespan	180	S12-1	11074	1	15	2.89
		S12-2	19586	1	36	1055.31
		S12-3	18862	1	42	8540.64
		S12-4	32523	1	78	18000.00
	30	S12-1	8524	1	15	3.14
		S12-2	15086	1	22	981.80
		S12-3	14462	1	30	3259.88
		S12-4	26580	1	49	5362.70

Table D.25. Lower Bounds for the Small/Medium Size Two–Machine Problems of Type 2 (Total Flow time)

Criteria	ASTH	Prob.	LB	Nnodes	Ncols	TT
TFT	180	S22-1	16746.00	1	9	1.45
		S22-4	47218.00	1	18	42.05
		S22-7	53867.00	1	15	3985.17
		S22-10	93556.00	1	71	11615.72
	30	S22-1	9996.00	1	11	2.66
		S22-4	31768.00	1	24	18000.00
		S22-7	37430.00	1	25	1958.20
		S22-10	68516.00	1	36	57.58
Makespan	180	S22-1	5755	1	14	2.89
		S22-4	10451	1	22	1055.31
		S22-7	10523	1	29	8540.64
		S22-10	14354	1	39	18000.00
	30	S22-1	3655	1	11	3.14
		S22-4	7601	1	21	981.80
		S22-7	7973	1	29	3259.88
		S22-10	11384	1	31	5362.70

Table D.26. Lower Bounds for the Large Size Two–Machine Problems of Type 1 (Total Flow time)

Criteria	ASTH	Prob.	LB	Nnodes	Ncols	TT
TFT	180	L12-1	613737	1	71	18000.00
		L12-2	615433	1	59	18000.00
		L12-3	785209	1	75	18000.00
		L12-4	985323	1	41	18000.00
	30	L12-1	497054	1	66	11026.21
		L12-2	484849	1	104	516.12
		L12-3	588707	1	96	18000.00
		L12-4	734062	1	105	18000.00
Makespan	180	L12-1	34181	1	74	18000.00
		L12-2	33924	1	81	18000.00
		L12-3	37435	1	99	18000.00
		L12-4	42124	1	116	18000.00
	30	L12-1	29033	1	68	168.53
		L12-2	28011	1	59	318.44
		L12-3	29857	1	102	18000.00
		L12-4	31906	1	119	18000.00

Table D.27. Lower Bounds for the Large Size Two–Machine Problems of Type 2 (Total Flow time)

Criteria	ASTH	Prob.	LB	Nnodes	Ncols	TT
TFT	180	L22-1	134441	1	76	11563.20
		L22-2	126942	1	21	18000.00
		L22-3	177330	1	65	18000.00
		L22-4	180709	1	73	18000.00
	30	L22-1	92999	1	104	42.33
		L22-2	90935	1	124	57.06
		L22-3	125022	1	136	10478.53
		L22-4	141555	1	130	1991.47
Makespan	180	L22-1	17487	1	55	18000.00
		L22-2	16121	1	67	18000.00
		L22-3	18132	1	237	18000.00
		L22-4	19068	1	99	18000.00
	30	L22-1	12949	1	188	9946.30
		L22-2	12760	1	216	8311.42
		L22-3	13742	1	193	18000.00
		L22-4	16002	1	301	8103.30

Table D.28. Lower Bounds for the Small/Medium Size Three–Machine Problems of Type 1 (Total Flow time)

Criteria	ASTH	Prob.	LB	Nnodes	Ncols	TT
TFT	180	S13-1	94669	1	18	3.39
		S13-2	223183	1	24	36.70
		S13-3	263532	11	119	18000.00
		S13-4	582255	9	83	18000.00
	30	S13-1	75958	1	14	2.48
		S13-2	167910	1	26	14.45
		S13-3	207353	1	63	43.09
		S13-4	490955	1	121	64.73
Makespan	180	S13-1	12937	1	31	5.83
		S13-2	19605	1	51	253.94
		S13-3	22356	1	72	7617.21
		S13-4	34274	1	81	18000.00
	30	S13-1	10401	1	30	165.32
		S13-2	15941	1	53	313.68
		S13-3	17406	1	67	2722.50
		S13-4	29578	1	91	18000.00

Table D.29. Lower Bounds for the Small/Medium Size Three–Machine Problems of Type 2 (Total Flow time)

Criteria	ASTH	Prob.	LB	Nnodes	Ncols	TT
TFT	180	S23-1	34945	1	30	2.28
		S23-2	25640	1	41	2.41
		S23-3	46618	1	59	3.44
		S23-4	113346	1	103	11.51
	30	S23-1	26995	1	29	1.35
		S23-2	18820	1	42	2.08
		S23-3	34549	1	59	2.36
		S23-4	88562	1	102	4.86
Makespan	180	S23-1	9026	1	36	1.78
		S23-2	6188	1	47	1.97
		S23-3	9297	1	67	6.29
		S23-4	17368	1	108	41.15
	30	S23-1	6806	1	39	2.27
		S23-2	4645	1	45	1.47
		S23-3	7353	1	64	5.90
		S23-4	14641	1	115	63.46

Table D.30. Lower Bounds for the Large Size Three–Machine Problems of Type 1 (Total Flow time)

Criteria	ASTH	Prob.	LB	Nnodes	Ncols	TT
TFT	180	L13-1	522520	1	47	18000.00
		L13-2	581288	1	88	18000.00
		L13-3	269454	1	165	18000.00
		L13-4	1071610	1	345	18000.00
	30	L13-1	394549	1	105	18000.00
		L13-2	473945	1	157	195.63
		L13-3	253189	1	112	18000.00
		L13-4	904125	1	145	13965.00
Makespan	180	L13-1	32186	1	125	18000.00
		L13-2	31587	1	89	18000.00
		L13-3	20381	1	72	18000.00
		L13-4	49152	1	138	18000.00
	30	L13-1	25176	1	61	18000.00
		L13-2	26247	1	74	18000.00
		L13-3	26841	1	90	18000.00
		L13-4	40118	1	117	18000.00

Table D.31. Lower Bounds for the Large Size Three–Machine Problems of Type 2 (Total Flow time)

Criteria	ASTH	Prob.	LB	Nnodes	Ncols	TT
TFT	180	L23-1	147343	1	157	2416.66
		L23-2	112281	1	91	18000.00
		L23-3	149620	1	121	18000.00
		L23-4	199193	1	148	18000.00
	30	L23-1	113246	1	100	21.41
		L23-2	96037	1	163	18000.00
		L23-3	112867	1	175	3683.30
		L23-4	166867	1	228	172.12
Makespan	180	L23-1	17966	1	79	13263.32
		L23-2	14817	1	80	18000.00
		L23-3	15971	1	114	18000.00
		L23-4	22334	1	142	18000.00
	30	L23-1	15145	1	95	2230.44
		L23-2	12174	1	76	18000.00
		L23-3	13309	1	129	12031.51
		L23-4	16857	1	113	18000.00

D.4. MILP1 and MILP3 Results

This section presents the results regarding the performance of the of MILP1 and MILP3 formulations. All small/medium size problems are solved with MILP1 and MILP3, all the large size problems are solved with MILP1. The description of the column headers in the tables are as follows.

Criteria : Minimizing the total flow time (TFT) or the makespan

ASTH : Average setup time per feeder on HSPM

N : The number of groups in the problem

BS : Best solution identified by the TS heuristics or MILP formulations

TTB : The time the best/optimal solution first identified, in seconds

TT : The total execution time, in seconds

Status : Status of the solutions identified by the MILP formulations

Optimal: An optimal solution is identified,

Feasible: A feasible solution is identified (optimality not verified)

Infeasible: A feasible solution could not be identified

%Gap : The percentage gap of MILP1/MILP3 solution from the global lower bound identified by CPLEX when it terminated

LP : LP–relaxation value of MILP1/MILP3

LPT : The time to solve LP–relaxation of MILP3, in seconds

When CPLEX could not identify even a feasible solution within the imposed time limit of five hours, a '-' sign is used to indicate that the corresponding value is not available.

Table D.32. MILP1 Results for 2-Machine Small/Medium Problems of Type 1

Criteria	ASTH	Prob.	N	BS	TTB	TT	Status	%Gap	LP	LPT
TFT	180	S12-1	3	65889	14.5	16.39	Optimal	0.00	56547.30	0.23
		S12-2	5	196548	1806.3	8010.45	Optimal	0.00	127002.00	3.77
		S12-3	6	219284	7968.1	9563.30	Optimal	0.00	172393.00	2.31
		S12-4	7	-	-	-	Infeasible	-	338779.1	13.30
	30	S12-1	3	47827	4.8	10.05	Optimal	0.00	44470.50	0.12
		S12-2	5	-	-	-	Infeasible	-	107927.00	1.24
		S12-3	6	169503	15206.3	18000.00	Feasible	5.12	145119.33	0.99
		S12-4	7	399810	9906.6	18000.00	Feasible	6.24	359096	1.24
Makespan	180	S12-1	3	11074	32.64	32.64	Optimal	0.00	8996.50	0.12
		S12-2	5	19594	2369.30	3619.77	Optimal	0.00	15211.31	3.42
		S12-3	6	19052	16207.90	18000.00	Feasible	7.27	14678.70	13.26
		S12-4	7	-	-	-	Infeasible	-	27010.90	37.08
	30	S12-1	3	8524	50.34	52.76	Optimal	0.00	7607.32	0.26
		S12-2	5	15094	3489.87	10112.00	Optimal	0.00	13089.90	3.88
		S12-3	6	-	-	18000.00	Infeasible	-	13266.58	8.92
		S12-4	7	-	-	18000.00	Infeasible	-	24067.57	32.46

Table D.33. MILP3 Results for 2–Machine Small/Medium Problems of Type 1

Criteria	ASTH	Prob.	N	BS	TTB	TT	Status	%Gap	LP	LPT
TFT	180	S12-1	3	65889	0.77	1.06	Optimal	0.00	47275.84	0.02
		S12-2	5	196548	4640.36	6911.14	Optimal	0.00	130842.05	0.05
		S12-3	6	219284	975.95	18000.00	Feasible	11.26	178336.02	0.12
		S12-4	7	502131	4704.82	18000.00	Feasible	15.05	391670.58	1.48
	30	S12-1	3	47827	0.31	1.17	Optimal	0.00	32811.84	0.02
		S12-2	5	140785	1456.83	18000.00	Feasible	7.91	96679.74	0.05
		S12-3	6	167573	3163.58	7603.09	Optimal	0.00	111538.32	0.12
		S12-4	7	400986	2488.63	18000.00	Feasible	11.21	318205.20	1.66
Makespan	180	S12-1	3	11074	0.17	0.28	Optimal	0.00	9559.66	0.03
		S12-2	5	19594	10.83	13.61	Optimal	0.00	17429.01	0.03
		S12-3	6	19052	22.36	23.42	Optimal	0.00	17342.04	0.11
		S12-4	7	33279	11.14	34.83	Optimal	0.00	29480.76	1.02
	30	S12-1	3	8524	0.20	0.33	Optimal	0.00	6983.31	0.02
		S12-2	5	15094	10.73	11.99	Optimal	0.00	12762.91	0.05
		S12-3	6	14702	29.48	29.85	Optimal	0.00	12900.10	0.11
		S12-4	7	27129	22.09	41.70	Optimal	0.00	23330.67	1.53

Table D.34. MILP1 Results for 2–Machine Small/Medium Problems of Type 2

Criteria	ASTH	Prob.	N	BS	TTB	TT	Status	%Gap	LP	LPT
TFT	180	S22-1	3	16746	0.55	0.56	Optimal	0.00	15927.6	0.06
		S22-2	5	47218	19.95	20.09	Optimal	0.00	35586.6	0.17
		S22-3	6	53867	11.64	20.05	Optimal	0.00	43443.7	0.14
		S22-4	8	94932	1658.89	17843.21	Optimal	0.00	70071.0	0.88
	30	S22-1	3	10274	0.47	0.52	Optimal	0.00	9806.5	0.05
		S22-2	5	32731	25.89	31.28	Optimal	0.00	29279.8	0.19
		S22-3	6	37754	9.81	9.98	Optimal	0.00	35836.5	0.19
		S22-4	8	69945	1941.92	5624.21	Optimal	0.00	60101.3	0.89
Makespan	180	S22-1	3	5830	0.50	0.56	Optimal	0.00	4606.2	0.06
		S22-2	5	10465	15.92	30.59	Optimal	0.00	7883.6	0.17
		S22-3	6	10591	53.62	100.53	Optimal	0.00	8525.2	0.19
		S22-4	8	14621	1118.46	1452.55	Optimal	0.00	11744.0	1.06
	30	S22-1	3	3685	0.66	0.69	Optimal	0.00	3496.2	0.05
		S22-2	5	7631	23.14	62.45	Optimal	0.00	7223.2	0.22
		S22-3	6	8003	4.03	81.86	Optimal	0.00	7664.8	0.17
		S22-4	8	11414	807.60	841.25	Optimal	0.00	10988.5	1.06

Table D.35. MILP3 Results for 2–Machine Small/Medium Problems of Type 2

Criteria	ASTH	Prob.	N	BS	TTB	TT	Status	%Gap	LP	LPT
TFT	180	S22-1	3	16746	0.06	0.08	Optimal	0.00	8056.20	0.01
		S22-2	5	47218	1.13	1.17	Optimal	0.00	27073.52	0.03
		S22-3	6	53867	4.23	7.66	Optimal	0.00	33536.34	0.11
		S22-4	8	94932	9707.91	10059.40	Optimal	0.00	55007.12	29.20
	30	S22-1	3	10274	0.08	0.09	Optimal	0.00	4624.30	0.02
		S22-2	5	32731	0.75	1.16	Optimal	0.00	17688.21	0.05
		S22-3	6	37754	7.70	8.94	Optimal	0.00	21916.44	0.11
		S22-4	8	69945	2816.32	3300.60	Optimal	0.00	41468.50	35.27
Makespan	180	S22-1	3	5830	0.09	0.09	Optimal	0.00	4918.10	0.01
		S22-2	5	10465	0. 22	0.30	Optimal	0.00	9201.74	0.03
		S22-3	6	10591	0.92	0.95	Optimal	0.00	9949.10	0.08
		S22-4	8	14621	544.43	544.56	Optimal	0.00	13586.23	24.48
	30	S22-1	3	3685	0.08	0.08	Optimal	0.00	2818.12	0.01
		S22-2	5	7631	0.19	0.25	Optimal	0.00	6201.78	0.03
		S22-3	6	8003	1.31	1.34	Optimal	0.00	7255.43	0.09
		S22-4	8	11414	335.37	336.46	Optimal	0.00	10392.37	25.20

Table D.36. MILP1 Results for 2–Machine Large Problems of Type 1

Criteria	ASTH	Prob.	N	BS	TTB	TT	Status	%Gap	LP	LPT
TFT	180	L12-1	9	-	-	18000	Infeasible	-	439666.51	315.12
		L12-2	10	-	-	18000	Infeasible	-	408154.70	210.90
		L12-3	11	-	-	18000	Infeasible	-	544285.00	778.85
		L12-4	12	-	-	18000	Infeasible	-	672993.16	1258.03
	30	L12-1	9	-	-	18000	Infeasible	-	410356.93	286.53
		L12-2	10	501256	11021.30	18000	Feasible	12.16	369509.22	175.36
		L12-3	11	-	-	18000	Infeasible	-	502926.04	1069.04
		L12-4	12	-	-	18000	Infeasible	-	631579.80	719.87
Makespan	180	L12-1	9	39156	5526.33	18000	Feasible	14.36	29699.7	260.20
		L12-2	10	-	-	18000	Infeasible	-	28887.41	236.64
		L12-3	11	-	-	18000	Infeasible	-	30697.03	1216.22
		L12-4	12	-	-	18000	Infeasible	-	35045.55	1595.03
	30	L12-1	9	-	-	18000	Infeasible	-	27066.72	247.56
		L12-2	10	-	-	18000	Infeasible	-	26051.16	255.72
		L12-3	11	-	-	18000	Infeasible	-	27939.98	1160.94
		L12-4	12	-	-	18000	Infeasible	-	31931.55	3362.01

Table D.37. MILP1 Results for 2–Machine Large Problems of Type 2

Criteria	ASTH	Prob.	N	BS	TTB	TT	Status	%Gap	LP	LPT
TFT	180	L22-1	10	146955	1909.54	18000.00	Feasible	17.04	100422.00	3.08
		L22-2	11	140833	5542.22	18000.00	Feasible	20.93	99547.30	5.59
		L22-3	13	246243	16513.40	18000.00	Feasible	44.45	127150.00	14.38
		L22-4	14	-	-	18000.00	Infeasible	-	138730.00	24.17
	30	L22-1	10	96851	3127.50	18000.00	Feasible	8.63	86180.60	3.52
		L22-2	11	94215	6594.06	18000.00	Feasible	7.41	83572.10	6.25
		L22-3	13	-	-	18000.00	Infeasible	-	112668.00	13.27
		L22-4	14	-	-	18000.00	Infeasible	-	123737.00	13.34
Makespan	180	L22-1	10	19318	13460.60	18000.00	Feasible	23.69	13495.30	4.31
		L22-2	11	17544	14852.70	18000.00	Feasible	19.45	13364.20	3.52
		L22-3	13	-	-	18000.00	Infeasible	-	14725.60	19.41
		L22-4	14	-	-	18000.00	Infeasible	-	15900.40	20.31
	30	L22-1	10	13968	10331.90	18000.00	Feasible	5.96	12128.65	6.13
		L22-2	11	13525	9935.26	18000.00	Feasible	5.22	11745.90	9.42
		L22-3	13	-	-	18000.00	Infeasible	-	13137.18	24.11
		L22-4	14	-	-	18000.00	Infeasible	-	14308.05	28.84

Table D.38. MILP1 Results for 3–Machine Small/Medium Problems of Type 1

Criteria	ASTH	Prob.	N	BS	TTB	TT	Status	%Gap	LP	LPT
TFT	180	S13-1	3	95559	97.59	108.61	Optimal	0.00	78182.00	0.63
		S13-2	5	-	-	18000.00	Infeasible	-	164280.00	10.58
		S13-3	6	-	-	18000.00	Infeasible	-	179211.00	15.88
		S13-4	8	-	-	18000.00	Infeasible	-	441961.00	139.29
	30	S13-1	3	77265	998.79	1211.20	Optimal	0.00	66628.70	0.78
		S13-2	5	-	-	18000.00	Infeasible	-	143530.00	10.64
		S13-3	6	-	-	18000.00	Infeasible	-	156981.00	15.11
		S13-4	8	-	-	18000.00	Infeasible	-	409901.00	141.25
Makespan	180	S13-1	3	13109	270.20	404.94	Optimal	0.00	10834.10	0.89
		S13-2	5	20207	10232.40	18000.00	Feasible	3.18	16297.40	9.08
		S13-3	6	-	-	18000.00	Infeasible	-	17606.80	13.48
		S13-4	8	-	-	18000.00	Infeasible	-	30908.60	187.20
	30	S13-1	3	10709	423.67	425.20	Optimal	0.00	8889.66	1.25
		S13-2	5	16569	2267.65	18000.00	Feasible	0.94	14662.39	15.16
		S13-3	6	-	-	18000.00	Infeasible	-	15023.16	16.47
		S13-4	8	-	-	18000.00	Infeasible	-	27680.27	122.87

Table D.39. MILP3 Results for 3–Machine Small/Medium Problems of Type 1

Criteria	ASTH	Prob.	N	BS	TTB	TT	Status	%Gap	LP	LPT
TFT	180	S13-1	3	95559	6.13	7.92	Optimal	0.00	77588.00	0.02
		S13-2	5	226068	7685.78	18000.00	Feasible	21.09	169659.00	0.05
		S13-3	6	279228	159.74	9782.93	Optimal	31.19	190426.00	0.17
		S13-4	8	644401	11203.60	18000.00	Feasible	33.86	421505.30	16.24
	30	S13-1	3	77265	15.86	22.83	Optimal	0.00	60660.40	0.02
		S13-2	5	176726	10719.10	18000.00	Feasible	40.12	124904.00	0.06
		S13-3	6	211829	442.53	941.79	Optimal	0.00	131542.00	0.22
		S13-4	8	509479	15081.00	18000.00	Feasible	39.51	319735.00	13.70
Makespan	180	S13-1	3	13109	2.47	3.48	Optimal	0.00	11021.40	0.02
		S13-2	5	19682	8.52	19.55	Optimal	0.00	17284.10	0.06
		S13-3	6	22476	112.64	126.83	Optimal	0.00	19868.50	0.13
		S13-4	8	36862	837.01	943.77	Optimal	0.00	34488.10	19.98
	30	S13-1	3	10709	3.17	3.47	Optimal	0.00	8816.00	0.02
		S13-2	5	16082	7.20	9.78	Optimal	0.00	13684.10	0.06
		S13-3	6	17526	4.59	7.70	Optimal	0.00	14735.40	0.16
		S13-4	8	31162	878.18	18000.00	Feasible	7.11	22435.90	101.30

Table D.40. MILP1 Results for 3–Machine Small/Medium Problems of Type 2

Criteria	ASTH	Prob.	N	BS	TTB	TT	Status	%Gap	LP	LPT
TFT	180	S23-1	4	35379	2.67	2.83	Optimal	0.00	31892.00	0.08
		S23-2	5	25946	7.81	8.08	Optimal	0.00	20575.80	0.11
		S23-3	6	46649	164.50	252.53	Optimal	0.00	38361.70	0.34
		S23-4	8	113442	9775.74	18000.00	Feasible	6.62	91855.70	1.22
	30	S23-1	4	27405	2.72	2.72	Optimal	0.00	26340.10	0.06
		S23-2	5	19182	7.56	7.84	Optimal	0.00	16358.30	0.08
		S23-3	6	35392	213.06	215.11	Optimal	0.00	32581.00	0.22
		S23-4	8	88894	6114.96	12909.51	Optimal	0.00	81640.70	1.27
Makespan	180	S23-1	4	9290	3.25	3.38	Optimal	0.00	7646.67	0.05
		S23-2	5	6312	11.44	14.03	Optimal	0.00	4938.90	0.08
		S23-3	6	9357	188.90	203.83	Optimal	0.00	7682.35	0.20
		S23-4	8	18116	9298.51	18000.00	Feasible	7.23	14994.40	1.56
	30	S23-1	4	6890	2.17	2.28	Optimal	0.00	6495.52	0.05
		S23-2	5	4782	14.98	17.25	Optimal	0.00	4001.65	0.05
		S23-3	6	7447	346.42	559.71	Optimal	0.00	6576.70	0.31
		S23-4	8	14886	8332.40	18000.00	Feasible	3.38	12810.15	1.74

Table D.41. MILP3 Results for 3–Machine Small/Medium Problems of Type 2

Criteria	ASTH	Prob.	N	BS	TTB	TT	Status	%Gap	LP	LPT
TFT	180	S23-1	4	35379	0.39	0.42	Optimal	0.00	20631.20	0.23
		S23-2	5	25946	0.54	0.87	Optimal	0.00	17659.30	0.36
		S23-3	6	46649	12.69	14.88	Optimal	0.00	25314.70	0.81
		S23-4	8	113442	1538.53	18000.00	Feasible	3.55	66619.60	38.92
	30	S23-1	4	27350	0.70	0.81	Optimal	0.00	14818.60	0.02
		S23-2	5	19182	0.97	0.99	Optimal	0.00	13644.00	0.03
		S23-3	6	35392	17.78	17.86	Optimal	0.00	18413.50	1.08
		S23-4	8	88894	5378.60	18000.00	Feasible	11.36	53913.80	68.22
Makespan	180	S23-1	4	9290	0.13	0.14	Optimal	0.00	8196.42	0.02
		S23-2	5	6312	0.69	0.70	Optimal	0.00	6363.76	0.03
		S23-3	6	9357	11.11	11.13	Optimal	0.00	8577.97	1.01
		S23-4	8	17616	1961.43	18000.00	Feasible	4.19	16698.00	58.47
	30	S23-1	4	6890	0.11	0.13	Optimal	0.00	5629.61	0.03
		S23-2	5	4782	0.98	1.33	Optimal	0.00	4334.28	0.05
		S23-3	6	7447	9.06	9.52	Optimal	0.00	6276.89	0.99
		S23-4	8	14704	2535.51	2559.98	Optimal	0.00	12540.12	91.20

Table D.42. MILP1 Results for 3–Machine Large Problems of Type 1

Criteria	ASTH	Prob.	N	BS	TTB	TT	Status	%Gap	LP	LPT
TFT	180	L13-1	9	-	-	18000.00	Infeasible	-	370507.00	214.46
		L13-2	10	665830	14231.30	18000.00	Feasible	39.43	400426.00	326.53
		L13-3	11	-	-	18000.00	Infeasible	-	77149.30	8.22
		L13-4	12	-	-	18000.00	Infeasible	-	772581.00	1513.60
	30	L13-1	9	-	-	18000.00	Infeasible	-	338471.00	232.63
		L13-2	10	499291	8860.09	18000.00	Feasible	-	376730.00	356.72
		L13-3	11	-	-	18000.00	Infeasible	-	68266.40	7.16
		L13-4	12	-	-	18000.00	Infeasible	-	731107.00	632.32
Makespan	180	L13-1	9	-	-	18000.00	Infeasible	-	27160.50	275.85
		L13-2	10	-	-	18000.00	Infeasible	-	27694.50	384.34
		L13-3	11	-	-	18000.00	Infeasible	-	9254.76	12.28
		L13-4	12	-	-	18000.00	Infeasible	-	40610.98	812.41
	30	L13-1	9	29872	15334.27	18000.00	Feasible	22.90	24497.13	329.23
		L13-2	10	-	-	18000.00	Infeasible	-	25150.92	363.91
		L13-3	11	-	-	18000.00	Infeasible	-	25220.93	174.63
		L13-4	12	-	-	18000.00	Infeasible	-	37092.12	405.11

Table D.43. MILP1 Results for 3–Machine Large Problems of Type 2

Criteria	ASTH	Prob.	N	BS	TTB	TT	Status	%Gap	LP	LPT
TFT	180	L23-1	10	-	-	18000.00	Infeasible	-	120236.90	3.66
		L23-2	11	-	-	18000.00	Infeasible	-	86619.60	12.28
		L23-3	12	-	-	18000.00	Infeasible	-	116345.12	14.91
		L23-4	14	-	-	18000.00	Infeasible	-	159830.76	31.14
	30	L23-1	10	117766	14131.79	18000.00	Feasible	5.88	107320.04	4.17
		L23-2	11	-	-	18000.00	Infeasible	-	78021.30	9.52
		L23-3	12	-	-	18000.00	Infeasible	-	101673.92	12.13
		L23-4	14	181165	7620.32	18000.00	Feasible	15.57	147123.00	30.73
Makespan	180	L23-1	10	-	-	18000.00	Infeasible	-	15772.20	3.69
		L23-2	11	-	-	18000.00	Infeasible	-	11611.10	13.23
		L23-3	12	-	-	18000.00	Infeasible	-	13459.35	13.17
		L23-4	14	-	-	18000.00	Infeasible	-	17601.52	50.34
	30	L23-1	10	-	-	18000.00	Infeasible	-	14033.82	4.74
		L23-2	11	-	-	18000.00	Infeasible	-	10324.41	18.72
		L23-3	12	-	-	18000.00	Infeasible	-	11992.09	28.08
		L23-4	14	-	-	18000.00	Infeasible	-	16058.20	65.41

APPENDIX E. A Large Size Real Industry Problem Data

The board groups and individual board types to be produced and their run times on the CP6 and IP3 machines are presented in Table E.44.

Table E.44. Run Times (in seconds) for the Large Size Industry Problem

Designated Group Name	Board Types in the Groups	Designated Name	Run Rate per Day	Run Time on CP6	Run Time on IP3
G_1	670-0624-05 Top	G_1	7	185	504
G_2	670-0914-01 Top	G_2	25	72	513
G_3	670-0915-03 Top	G_3	25	576	2057
G_4	670-0878-04 Top	G_{41}	25	721	706
	670-0878-05 Top	G_{42}	25	865	1000
	670-0878-07 Top	G_{43}	25	432	1092
G_5	670-1297-01 Top	G_5	15	373	258
G_6	670-1298-00 Top	G_6	15	395	1137
G_7	670-0624-05 Bot	G_7	7	319	341
G_8	670-0914-01 Bot	G_8	25	1398	130
G_9	670-0915-03 Bot	G_9	25	325	412
G_{10}	670-1297-01 Bot	G_{10}	15	669	696
G_{11}	670-1298-00 Bot	G_{11}	15	139	262

Tables E.45 and E.46 present the component–feeder configurations of the board types on the CP6 and IP3 machines, respectively.

Table E.45. Feeder Configuration for the CP6 Machine

Feeder	G_1	G_2	G_3	G_{41}	G_{42}	G_{43}
1	SLM150-0041-00	SLM297-0120-00			SLM327-0456-00	SLM327-0456-00
2	SLM297-0169-00	SLM327-0060-00	SLM276-0072-00	SLM326-0181-00	SLM327-0110-00	SLM327-0110-00
3	SLM297-0111-00	SLM297-0159-00			SLM153-0001-00	SLM153-0001-00
4	SLM327-0050-00		SLM297-0526-00	SLM327-0818-00	SLM276-0042-00	SLM276-0042-00
5	SLM327-0110-00	SLM150-0041-00	SLM327-0105-00		SLM327-0668-00	
6			SLM327-0110-00	SLM326-0198-00	SLM326-0198-00	SLM326-0175-00
7	SLM296-0013-00	SLM326-0150-00	SLM327-0246-00	SLM171-0102-00	SLM276-0034-00	
8			SLM327-0161-00	SLM327-0161-00	SLM148-0019-00	SLM148-0019-00
9		SLM296-0069-00	SLM327-0503-00	SLM327-0503-00	SLM326-0198-00	SLM326-0198-00
10			SLM327-0286-00	SLM327-0286-00	SLM171-0102-00	SLM171-0102-00
11			SLM297-0169-00	SLM297-0111-00	SLM327-0677-00	SLM155-0019-00
12			SLM297-0284-00	SLM155-0057-00	SLM155-0057-00	SLM297-0022-00
13			SLM327-0461-00	SLM296-0100-00	SLM296-0100-00	
14			SLM153-0013-00	SLM276-0034-00	SLM155-0057-00	
15			SLM155-0057-00	SLM276-0042-00		
16				SLM297-0022-00	SLM297-0022-00	SLM297-0183-00
17			SLM150-0041-00			
18				SLM297-0183-00	SLM296-0031-00	SLM296-0031-00
19			SLM155-0129-00			
20				SLM296-0031-00	SLM327-0682-00	SLM327-0682-00

Table E.45. Feeder Configuration for the CP6 Machine (Continued)

Feeder	G_5	G_6	G_7	G_8
1	SLM297-0120-00	SLM297-0284-00	SLM296-0064-00	SLM297-0012-00
2	SLM327-0060-00	SLM297-0526-00	SLM153-0013-00	SLM327-0086-00
3	SLM297-0159-00		SLM297-0205-00	SLM297-0120-00
4		SLM150-0041-00	SLM327-0050-00	SLM297-0159-00
5	SLM327-0105-00		SLM297-0221-00	SLM297-0055-00
6		SLM327-1388-00	SLM327-0110-00	SLM327-0286-00
7	SLM326-0150-00	SLM327-1417-00	SLM297-0245-00	SLM327-0060-00
8		SLM327-1484-00		SLM170-0049-00
9	SLM327-0503-00	SLM296-0031-00	SLM172-0149-00	SLM172-0343-00
10		SLM327-1384-00		SLM172-0429-00
11		SLM327-0050-00	SLM296-0013-00	SLM297-0120-00
12		SLM327-1571-00		SLM297-0011-00
13		SLM327-0461-00	SLM327-1266-00	SLM172-0049-00
14		SLM297-0250-00		
15		SLM297-0203-00	SLM326-0168-00	SLM150-0041-00
16		SLM327-0161-00		
17		SLM327-0105-00		SLM326-0150-00
18		SLM327-0246-00		
19		SLM327-0503-00		SLM150-0094-00
20		SLM327-0110-00		SLM297-0500-00

Table E.45. Feeder Configuration for the CP3 Machine (Continued)

Feeder	G_9	G_{10}	G_{11}
1	SLM297-0284-00	SLM297-0284-00	SLM297-0159-00
2	SLM327-0261-00	SLM327-0261-00	SLM297-0012-00
3			SLM327-0086-00
4	SLM150-0041-00	SLM150-0041-00	SLM172-0343-00
5			SLM170-0049-00
6			SLM327-0286-00
7		SLM327-0060-00	SLM327-0060-00
8			SLM297-0120-00
9			SLM297-0500-00
10			SLM297-0120-00
11			SLM172-0429-00
12			
13			SLM326-0150-00
14			
15			SLM150-0094-00
16			SLM172-0049-00
17			
18			
19			
20			

Table E.46. Feeder Configuration for the IP3 Machine

Feeder	G_1	G_2	G_3	G_{41}	G_{42}	G_{43}
1		SLM171-0275-00			SLM171-0282-00	
2		SLM155-0112-00	SLM171-0281-00			
3				SLM148-0019-00	SLM148-0019-00	
4	SLM172-0516-00	SLM214-0522-00	SLM299-0028-00			
5				SLM276-0055-00	SLM172-0149-00	
6	SLM158-0088-00	SLM214-0523-00	SLM108-0175-00	SLM276-0006-00	SLM276-0006-00	SLM276-0006-00
7	SLM135-0013-00	SLM108-0175-00	SLM299-0026-00	SLM171-0345-00	SLM276-0055-00	SLM276-0055-00
8	SLM174-0280-00	SLM171-0215-00	SLM299-0029-00	SLM172-0149-00	SLM171-0345-00	
9		SLM299-0006-00	SLM153-0073-00	SLM171-0282-00	SLM171-0010-00	
10				SLM171-0010-00		

Table E.46. Feeder Configuration for the IP3 Machine (Continued)

Feeder	G_5	G_6	G_7	G_8
1		SLM108-0175-00		
2				
3		SLM171-0281-00		
4				
5		SLM299-0028-00		SLM155-0117-00
6	SLM299-0006-00	SLM155-0117-00	SLM170-0068-00	SLM155-0112-00
7	SLM155-0112-00	SLM299-0026-00	SLM172-0276-00	SLM171-0276-00
8	SLM171-0275-00	SLM299-0029-00	SLM173-0057-00	
9		SLM153-0073-00		
10	SLM214-0522-00			

Table E.46. Feeder Configuration for the IP3 Machine (Continued)

Feeder	G_9	G_{10}	G_{11}
1			
2			
3			
4			
5		SLM155-0117-00	SLM171-0216-00
6	SLM155-0117-00	SLM171-0276-00	SLM155-0117-00
7	SLM171-0215-00	SLM155-0112-00	SLM171-0215-00
8	SLM171-0214-00		SLM171-0214-00
9	SLM171-0216-00		
10			