AN ABSTRACT OF THE THESIS OF

Vaibhav Pandya for the degree of Master of Science in Industrial Engineering presented on June 15, 2011.

Title: A Methodology for Scheduling Jobs in a Flexible Flowshop with Sequence Dependent Setup Times and Machine Skipping.

Abstract approved: ____

Rasaratnam Logendran

A flexible flowshop, comprised of one or more stages having unrelated parallel machines, is investigated in this research. Unrelated parallel machines can perform the same function but have different capacity or capability. Since this problem is motivated by industry research, dynamic job release times and dynamic machine availability times have been considered. Each job considered in this research can have different weight and due date. Sequence-dependent setup times of jobs further add to the complexity of the research. Machine skipping is yet another innate feature of this research that allows jobs to skip one or more stages depending upon customer's demand or budgetary constraints. The objective of this research is to minimize the sum of the weighted tardiness of all jobs released within the planning horizon.

The research problem is modeled as a mixed (binary) integer-linear programming model and is shown to be strictly NP-hard. Being strongly NP-hard, industry size problems cannot be solved using an implicit enumeration technique within a reasonable computation time. Cognizant of the challenges in solving industry-size problems, we use the tabu-search-based heuristic solution algorithm to find optimal/near optimal solutions. Five different initial solution finding mechanisms, based on dispatching rules, have been developed, to initiate the search. The five initial solution finding mechanisms (IS1-IS5) have been used in conjunction with the six tabu-search-based heuristics (TS1-TS6) to solve the problems in an effective and efficient manner. The applicability of the search algorithm on an example problem has been demonstrated. The tabu-search based heuristics are tested on seven small problems and the quality of their solutions is compared to the optimal solutions obtained by the branch-and-bound technique. The evaluations show that the tabu-search based heuristics are capable of obtaining solutions of good quality within a much shorter computation time. The best performer among these heuristics recorded a percentage deviation of only 2.19%.

The performance of the tabu-search based heuristics is compared by conducting a statistical experiment that is based on a randomized complete block design. Three sizes of problem structures ranging from 9 jobs to 55 jobs are used in the experiment. The results of the experiment suggest that no IS finding mechanism or TS algorithm contributed to identifying a better quality solution (i.e a lower TWT) for all three problem instances (i.e. small, medium and large). In other words, no IS finding mechanism or TS algorithm could statistically outperform others. In absence of a distinct outperformer, TS1 with short-term memory and fixed TLS are recommended for all problem instances. When comparing the efficiency of the search algorithm, the results of the experiment show that IS1, which refers to the EDD (earliest due date) method, is recommended as the initial solution generation method for small problem sizes. The EDD method is capable of obtaining an initial solution that helps the tabu-search based heuristic to get to the final solution within a short time. TS1 is recommended as the tabu-search based heuristic for large problems, in order to save on time. TS1 is also recommended to solve small and medium problem structures in absence of a statistically proven outperformer.

A Methodology for Scheduling Jobs in a Flexible Flowshop with Sequence Dependent Setup Times and Machine Skipping

by

Vaibhav Pandya

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented June 15, 2011
Commencement June 2012

Master of Science thesis of Vaibhav Pandya presented on June 15, 2011

APPROVED:

Major Professor, representing Industrial Engineering

Head of the School of Mechanical, Industrial and Manufacturing Engineering

Dean of the Graduate School

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

Vaibhav Pandya, Author

ACKNOWLEDGEMENTS

This thesis wouldn’t have come to fruition without the support of my dear ones. They have provided me with unconditional help and encouragement at every stage of my life. I will be indebted to them for the rest of my life.

I would like to convey my sincere gratitude to Dr. Logendran, my Major Professor, who gave me the opportunity to work on a problem that was industry funded. He helped me on each and every step and was extremely patient throughout my stay at Oregon State University. This funded project was my first opportunity to work with industry professionals on a day to day basis. This project prepared me to face challenges that are prevalent in the industry.

I would like to thank my employer Microchip Technologies for being exceptionally cooperative. Their help and support was phenomenal. I would like to thank my committee members: Dr. Dan Schafer, my minor professor, Dr. Karl Haapala, my committee member, and Dr. Mario Magana, my graduate council representative, for taking the time to evaluate my thesis.

My friends at OSU, SWOSU and India made my journey remarkably effortless. I would like to thank Amelia, Chetan, Ravi, Tulsi, Valmiki, Venkatesh, and Vineeth for all their help and support. Raul for helping me out with statistical analysis. I would also like to thank my Persian friends Amir, Yasaman and Hadi for helping me out with my course work and thesis. Pankaj’s and Ziggy’s friendship has been invaluable and their support has been one of my biggest strengths. So thank you everyone for being there.

Finally, I would like to express my sincere gratefulness to my parents, my beloved sisters Vartika and Shweta, and my grand mom (dadi) for their prayers, support and encouragement. My Grand dad whose blessings have helped me conquer new heights. Pooja for standing by my side during my rough phase and for being extremely understanding and patient. My teachers at OSU, SWOSU and India, whose priceless contributions have made things look incredibly easy. And everyone back home, who made this journey successful. Thank you once again.

TABLE OF CONTENTS

Page

1 INTRODUCTION 1

2 LITERATURE REVIEW 4

3 PROBLEM STATEMENT 9

4 MODEL DEVELOPMENT 11

4.1 Introduction 11

4.2 Assumptions 11

4.3 Notations: 12

4.4 Mathematical Model 13

4.5 Model Description 14

4.6 Computational Complexity of the Research Problem 15

5 HEURISTIC ALGORITHM 17

5.1 Introduction 17

5.2 Tabu Search Mechanism 18

5.3 Initial Solution 21

5.4 Generation of Neighborhood Solutions 36

5.5 Steps of Tabu Search 38

5.6 Application of Heuristic Algorithm to Example Problem 45

6 THE OPTIMALITY OF TABU-SEARCH BASED HEURISTIC ALGORITHM .. 81

6.1 Comparison Between the Optimal Solution and Solution Obtained by the Heuristic Algorithm 83

6.2 The Effectiveness of Tabu-Search Based Heuristics for Medium and Large Problems 87

7 RESULTS AND DISCUSSION 93

7.1 Data Generation 96

7.2 Design of Experiment 100

7.3 Experimental Results and Analysis 103

8 Conclusion and Suggestions for further research 117

BIBLIOGRAPHY 121

APPENDICES 126

LIST OF TABLES

Table Page

Table 5-1 Problem Structure 46

Table 5-2 Example problem with 11 jobs 46

Table 5-3 Initial solutions of example problem 62

Table 5-4 The neighborhood solutions of initial solution as a result of applying swap and insert moves 63

Table 5-5 Entries into the LTM matrix after perturbing the initial solution 68

Table 5-6 Results of tabu search applied to the initial solution of the example problem. 69

Table 5-7 Entries into the LTM matrix at the end of the search using the initial solution71

Table 5-8 Results from the first restart based on maximal frequency 72

Table 5-9 Entries into the LTM matrix at the end of the first restart based on maximum frequency 73

Table 5-10 Results from the second restart based on maximal frequency 74

Table 5-11 Summary of results for the entire search process based on LTM-max 75

Table 5-12 Results of first restart based on minimum frequency 76

Table 5-13 Entries into the LTM matix at the end of first restart based on minimum frequency 77

Table 5-14 Results of second restart based on minimum frequency 78

Table 5-15 Summary of results for the entire search process based on LTM-min 80

Table 6-1 Results of solving the problems implicitly using CPLEX 9.0 83

Table 6-2 Tabu search based heuristic algorithms used in this research 84

Table 6-3 Percentage deviation of the solutions obtained by the heuristics for small problems 84

Table 6-4 Computation time of the heuristics for small problems (in seconds) 86

Table 6-6 Results of applying the heuristics to medium problem structures with zero values of TWT 88

LIST OF TABLES (Continued)

Table Page

Table 6-7 Average percentage deviation of the solutions obtained by the heuristics for medium problem structure 90

Table 6-8 Results of applying the heuristics to large problem structures with zero values of TWT........ 91

Table 6-9 Average percentage deviation of the solutions obtained by the heuristics for large problem structure 92

Table 7-1 Due date classification........ 99

Table 7-2 Results for small size problems from ANOVA for TWT 106

Table 7-3 Results for small size problems from ANOVA for CT........ 107

Table 7-4 Differences of least square means for CT of small size problems (IS)........ 107

Table 7-5 Results for medium size problems from ANOVA for TWT........ 109

Table 7-6 Results for medium size problems from ANOVA for CT 111

Table 7-7 Differences of least square means for CT for medium size problems 111

Table 7-8 Results for large size problems from ANOVA for TWT........ 113

Table 7-9 Results for large size problems from ANOVA for CT........ 114

Table 7-10 Differences of least squares means for CT of large size problems 115

DEDICATION

To my late grandfather - Shri Vinay Shanker Pandya

A Methodology for Scheduling Jobs in a Flexible Flowshop with Sequence Dependent Setup Times and Machine Skipping

1 INTRODUCTION

Manufacturing firms are persistently discovering ingenious techniques to overcome the fierce completion from their counterparts. Conventional manufacturing approaches are constantly being replaced by novel practices to improve the overall efficiency and effectiveness of the manufacturing system. Various manufacturing firms are resorting to scheduling algorithms to help them meet the customer requirements in a timely manner, and to reduce their operational costs.

The two vital components of a scheduling problem are machine configurations and job characteristics. Typically, the machine configuration of a manufacturing plant consists of either a flow shop or a job shop setting, with single or multiple machines. A flow shop setting or a job shop setting may consist of several stages. Typically, in a flow shop setting, each stage has only a single machine. If at least one stage of the flow shop has more than one machine, it is termed as a flexible flow shop. Characteristically, in a flow shop setting, jobs follow a flow line and have exactly one operation on each machine. On the other hand, in a job shop setting, jobs do not strictly adhere to a flow line.

The machines in a flexible flow shop environment can be classified into three categories: identical parallel machines, uniform parallel machines and unrelated parallel machines. In an identical parallel machine system, the processing time of a job is the same on all machines. In a uniform parallel machine system, each machine has a unique speed factor. The processing time of a job on each machine is determined by its speed factor. In unrelated parallel machines, the processing time of a job may differ from one machine to the other, depending on machine's capacity (low, medium or high). In an unrelated parallel machines system, a machine with low capability usually takes more time to process a job than a machine with high capability. Unrelated parallel machines system is extensively used in the industry because a company may invest in similar machines with different capability to cut down on investment or capital costs or to meet the variation in production demand.

As mentioned earlier, along with machine configurations, job characteristics are also an innate component of a scheduling problem. Cheng and Sin (1990) listed five characteristics of a job: job processing time, due-date, preemptive sequencing, precedence constraints and job release time. The first two characteristics do not require much explanation and are relatively easy to understand. The third characteristic (preemptive sequencing) refers to the idea of giving a particular job precedence over the other, if it is deemed urgent. The fourth characteristic (precedence constraints) determines the sequence in which the jobs are supposed to be processed on each machine. The fifth characteristic (release time) of jobs refers to the time when a particular job is available on a shop floor. A scenario in which one or more jobs are released at different times is termed as dynamic job availability. On the other hand, if all the jobs are available to be processed at the same time, it is called static job availability. Lodree et al. (2004) suggested a scheduling rule for minimizing the number of tardy jobs in a dynamic flow shop environment, consisting of *m* machines. The processing times and due dates of jobs were allotted randomly. The research assumed that the availability time of jobs was not known in advance and that the jobs arrive randomly to the system.

Similar to jobs, machines may also have different release times. If machines are released at different times, the condition is called dynamic machine availability and if all the machines are released at the same time, it is called static machine availability. In an industrial environment, jobs compete for limited resources based on customer’s priority and due-date. A job which has a high priority and tight due date must be given preference over another job with low priority and loose due date. A superior scheduling algorithm must be capable of providing a sequence of jobs that would not only reduce the manufacturing costs, but also meet the customer demand on time.

Since this research is aimed at solving a problem that is very much prevalent in industry, jobs are assumed to have sequence-dependent setup times on machines. Setup time may be defined as the time required for changing over from one job to another, on a particular machine. If the setup time varies when changing over from one job to another, then the jobs are said to have sequence-dependent setup times. If the setup time remains the same for changing over from one job to another, then the jobs are said to have sequence-independent setup times. Tasgetiren (2009) investigated the possibilities of minimizing the total weighted tardiness of all jobs on a single machine with sequence-dependent setup times. Pfund et al. (2008) explored the possibilities of scheduling jobs of on identical parallel machines with sequence-dependent setup times. The purpose of their study was to minimize the weighted tardiness of all the jobs.

The problem investigated in this thesis is directly motivated by a leading manufacturing company. The industrial setting considered in the research problem is that of a flexible flowshop with the possibility of machine skipping. A flexible flowshop comprises of multiple stages with at least one (or more) stage having more than one machine. The scheduling environment considered in this research is assumed to be dynamic in job release time as well as machine availability time. The purpose of this research is to find an optimal or near optimal schedule that would minimize the weighted tardiness of all the jobs. Such an attempt is very much relevant to the industry since customer satisfaction is the primary concern of all the firms. A job in this research can be viewed as a customer order that has a ‘strategic weight’ associated with it. The weight of the job reflects its priority, i.e. a job with higher weight receives greater priority. The tardiness of a job is evaluated as the difference in completion time and the due date of a job. A negative tardiness is considered to be zero and suggests that the job was completed before the due date. Weighed tardiness is evaluated by multiplying tardiness of a job by its weight.

2 LITERATURE REVIEW

Production scheduling has been a key area of interest for several researchers over the past 50 years. Johnson (1954) was the first one to introduce a systematic approach for finding a mechanism that helped in obtaining an optimal solution for a two machine flow shop (and also for a special case of three machines). Campbell, Dudek and Smith (CDS) (1970) proposed a methodology that was an extension of Johnson's Algorithm. CDS methodology uses the Johnson's algorithm in a heuristic manner whereby several schedules are created from which the best is chosen. Nawaz, Enscore, and Ham (1983) proposed a new algorithm based on the assumption that the job with greater total processing time on all machines should be given priority over jobs with a lower total processing time. Unlike the CDS algorithm, the NEH algorithm doesn't transform a *m*-machine problem into a two machine problem. Instead, it aims at generating a best partial solution for the original problem by adding a job in each step, to finally identify the best (complete) solution.

Sriskandarajah and Sethi (1989) proposed a heuristic algorithm for a unique flexible flow-shop problem, with only two machine centers. They assumed that each machine center had the same number of homogeneous machines. Their research had an underlying supposition that the primary operation on each job must initially be performed on the first machine, followed by the second. The jobs were processed on any vacant machine at the machine center, depending on its availability. Kyparisis and Koulamas (2006) also investigated a two-stage flexible flow shop problem with uniform parallel machines. The primary purpose of their research was to minimize the makespan of all the jobs.

Taking advantage of branch-and-bound algorithm, Salvador (1973) investigated a flexible flow shop problem aimed at minimizing the makespan with no in-process buffers. A branch-and-bound algorithm for the general flow shop, with parallel machines, was investigated by Hunsucker (1991). This structure became popular by the name of hybrid flow shops and included in-process buffers. However, the algorithm was incapable of solving large combinatorial problems, revealing the limitations of mixed integer programming algorithms. Ding and Kittichartphayak (1994) developed three different heuristic algorithms for minimizing the makespan in a flexible flow line. Their algorithm was an extension of the heuristic algorithms used in regular flow shop scheduling problems.

Jaymohan and Rajendran (2000) explored the relative effectiveness of two approaches to schedule jobs in a flexible flow shop environment. The first approach utilized the possibility of using different dispatching rules at different stages of the flow shop and the second approach utilized the same dispatching rule for all the stages in the flow-shop. Both approaches were aimed at minimizing the flow time and tardiness of jobs. The authors argue that most dispatching rules for flow shop scheduling assume that the cost of holding per unit time is the same for all jobs. The authors further say that it is assumed that the cost of tardiness per unit time is the same for all jobs. In other words, it is implied that the holding cost of a job is directly proportional to its flowtime, and the tardiness cost of a job is directly proportional to its positive lateness. The authors try to overcome this deficiency by proposing a couple of dispatching rules, which consider different weights or penalties for different jobs.

Wang (2005) investigated several scholarly articles published on flexible flow shop scheduling. The purpose of the research was to review all the solution approaches adopted in flexible flow shop scheduling, ranging from optimum to heuristic solutions. The article not only presents a detail analysis of all the approaches, but also provides suggestions for future research.

Flexible flow shop scheduling problems have held the interest of researchers primarily because of its industrial relevance. Kurz and Askin (2003) investigated scheduling of jobs in flexible flow lines with sequence-dependent setup times. They explored heuristic approaches including the insertion heuristic, Johnson’s algorithm for two-stage flow shops and its heuristic extensions to *m*-machine flow shops. Furthermore, Kurz and Askin (2004) developed a random keys genetic algorithm for the problem they had investigated in (Kurz and Askin, 2003), primarily to evaluate their proposed random key genetic algorithm against other heuristics they had proposed earlier. With the objective of minimizing the make-span, Zandieh et al. (2006) investigated the same problem and proposed an immune algorithm. They successfully demonstrated that the immune algorithm outperformed the random key genetic algorithm proposed by Kurz and Askin (2004).

Allahverdi et al. (2006) completed a survey of all scheduling problems with setup times. Naderi, Ruiz, and Zandieh (2009) investigated a flowshop problem with parallel machines and sequence dependent setup times where the objective was to minimize the makespan. They made an assumption that all jobs need not visit each and every stage, meaning that machine skipping was allowed. They introduced a heuristic in the form of a dynamic dispatching rule that is based on iterated local search.

Jungwattanakit et al. (2009) considered a flexible flow shop scheduling problem, taking sequence- and machine-dependent setup times into consideration. The study aimed at developing a bi-criteria model for minimizing the convex sum of makespan and the number of tardy jobs in a static flexible flow shop setting. The authors initially formulated a mixed integer programming model to find the optimal solution for an industry size problem. But due to large computation time associated with running the mixed integer programming model, they suggested other heuristic approaches to obtain a quality solution within a reasonable time. The authors primarily used metaheuristic algorithms such as simulated annealing (SA), tabu search (TS) and genetic algorithms to obtain an optimal or near optimal solution.

Naderi et al. (2009) considered a flexible flowshop scheduling problem with sequence dependent set-up time and job-independent transportation times. The primary objective of the research was to minimize the total weighted tardiness. However, they did not consider dynamic release of jobs and dynamic machine availability, which are commonly prevalent in industry situations. All the machines in a given stage are considered to be uniform parallel. Furthermore, machine skipping is not permitted in the problem they investigated. They utilize electromagnetic algorithm (EMA) to efficiently solve the research problem. EMA is a meta-heuristic algorithm that originated from the attraction-repulsion mechanism of the electromagnetism theory. The authors also formulated a mixed integer linear programming (MILP) model for optimally solving small-sized problem instances.

A large number of research efforts have been made in the past based on tardiness objectives in a flowshop. To minimize mean tardiness, Kim (1993) modified several priority rules including earliest due date (EDD), minimum slack (SLACK), slack per remaining work (SRMWK) and modified due date (MDD). The author also demonstrated the utilization of tabu search to solve the problem using EDD as initial solution. Rajendran (1997) utilized simulated annealing (a probabilistic meta-heuristic search algorithm for obtaining global maxima or minima) for a sequence dependent flexible flow shop problem to minimize the maximum weighted tardiness and the total weighted tardiness.

Vallada et al. (2008) performed a comprehensive evaluation of heuristics and metaheuristics for the *m*-machine flowshop scheduling problem with the objective of minimizing total tardiness. Fourty different heuristics and metaheuristics were implemented and their performance was analyzed using same benchmark of instances in order to make a fair comparison. The study entails a wide variety of classical priority rules as well as most recent metaheuristic algorithms such as tabu search, simulated annealing and genetic algorithms. An experimental design approach was used by the researchers to carry out the statistical analysis and to validate the effectiveness of the different methods tested.

In the past few years, tabu search has proved to be remarkably successful at solving notoriously complex problems of industrial merit. Tabu search was first introduced by Glover (1996). A synopsis of the application of tabu search on production scheduling problem was investigated by Barnes et al. (1995). The study listed tabu search based applications in a single-machines problem, parallel machines problems, travelling salesman problem, flow shop problem, vehicle routing problem, classical job shop problem and flexible job-shop problem. Muller et al. (1995) investigated the relevance of tabu search in solving identical parallel machine scheduling problem with sequence dependent setup times, to minimize the makespan. The algorithm consists of three phases: initial assignment, tabu search, and post-optimization procedure.

Logendran and Subur (2004) utilized the tabu search based heuristics to schedule jobs on unrelated parallel machines, with the possibility of job splitting. Their research incorporated various tabu search features and led to the development of six different heuristic algorithms. The study showed that the proposed heuristic algorithms were capable of solving large complex problems in surprisingly short time. A mathematical model for the problem was also developed to demonstrate that the tabu search solutions identified were very effective by comparing them to the optimal solutions obtained for small problem instances.

Logendran et al. (2006) suggested a heuristic approach for solving sequence dependent group scheduling problems of industrial merit utilizing tabu search. The objective of the research was to minimize the makespan required to process all the jobs in all the groups that were released on the shop floor. Their research took into consideration of problem sizes ranging from small, medium to large, and suggested a methodology for solving large problems in an efficient and effective manner. Logendran et al. (2007) employed the tabu search based heuristics to schedule jobs on unrelated parallel machines with sequence-dependent setup times. They proposed six different search algorithms based on tabu search heuristics to minimize the total weighted tardiness.

To summarize, the scheduling environment of this research is dynamic in both job release time and machine availability. The objective of the research focuses on finding optimal/near-optimal schedule that minimizes the sum of the weighted tardiness of all jobs in a flexible flowshop environment. Such an objective is important in many industrial applications since on-time delivery is one of the most important factors in customer satisfaction. Machine skipping is another important feature in this research. A job can skip one or more stages depending upon customer’s request. Additionally, each job has a strategic weight associated with it, which reflects its priority, i.e. job with higher priority receives higher weight.

3 PROBLEM STATEMENT

This research focuses on scheduling jobs in a flexible flow shop with sequence-dependent setup times. Since it is a flexible flow shop scheduling problem, one or more stages may have more than one machine. The stage with multiple machines may comprise of identical, uniform or unrelated parallel machines. If a stage comprises of unrelated parallel machines, not all jobs would be eligible to be processed on all machines. The release times of jobs and the availability times of machines are assumed to be dynamic. It means that the jobs can be released at any time during the current planning horizon, depending upon the customer's request or order. Similarly, at the start of the current planning horizon, a subset of machines might be unavailable due to processing jobs from the previous planning horizon, only to become available at a later time. These assumptions are in conjunction with what is typically observed in an industrial setting. Machine skipping is an innate feature of this research problem. A job may skip one or more stages depending upon customer's requirement or budgetary constraints. In other words, if a customer deems a particular process as a non value adding activity, he may opt not to perform it. The possibility of a job skipping a stage makes this problem even more intricate.

Companies usually associate a project with a unique customer so that they can make a clear distinction among customers. The jobs considered in this paper are directly associated with a particular project, and that association is an integral part of this research. Consequently, a project may consist of one or more jobs, depending upon customer requirements. Thus, a project may be considered as a larger entity for distinguishing customers and the jobs can be considered as smaller components or elemental units associated with a project. Additionally, the jobs within the same project may be similar or dissimilar depending upon customer's requirements. Therefore, the jobs within a project may have an entirely different run time on the same machine. There is also a sequence-dependent setup time associated with changing over from one job to another, on a particular machine. Similar jobs will have smaller setup times, while dissimilar jobs will have larger setup times.

The objectives of this research are as follows:

i. To develop a mathematical model for minimizing the total weighted tardiness of jobs which are dynamically released in a flexible flow shop, with the possibility of machine skipping. (Note that jobs considered in this research have sequence-dependent setup times and the machine availability is considered to be dynamic)

ii. To develop a scheduling algorithm that would efficiently solve the mathematical model considered in (i).

4 MODEL DEVELOPMENT

4.1 Introduction

A mixed (binary) integer linear programming model that captures the operational constraints of an industrial setting is developed to quantify the effectiveness of the proposed solution algorithm. The parameters used in the model such as the number of jobs, number of stages, number of machines in a given stage, job release times, machine availability times, job weights, job due dates, job run times, sequence-dependent setup times, and whether or not a job has an operation in a given stage, are known quantities.

Notice that during a planning horizon, a machine may become unavailable due to an unanticipated breakdown. It is extremely difficult to predict when the machine might break down and how long it will remain unavailable. Moreover, introducing additional constraints add to the complexity of an already complex model. Therefore, for the purpose of modeling, we assume that during a given planning horizon, a machine remains fully functional from the time it becomes available.

4.2 Assumptions

(1) Each job can be processed only once at each stage

(2) A job may skip one or more stages depending upon customer requirements

(3) Even though a job may skip one or more stages, it still follows a flow line (unidirectional) arrangement

(4) Cross precedence is not allowed, meaning that a job cannot be at the same time both a predecessor and a successor of another job

(5) Sequence-dependent setup times of jobs are known

(6) No preemption is allowed

(7) Each machine can process only one job at a time

(8) Not all jobs can be processed on all machines. If a job cannot be processed on a machine, it is assumed to have a very large run time on that machine

4.3 Notations:

$j = 1, \dots, n$ jobs

$g = 1, \dots, G$ stages

$i = 1, \dots, m_g$ number of machines in stage g

Parameters

a_{ig} = availability time of machine i in stage g

r_{ijg} = run time of job j on machine i in stage g

w_j = weight assigned to job j

d_j = due date of job j

e_{ijg} = release time of job j on machine i in stage g

s_{ikjg} = setup time for job j, immediately following job k on machine i of stage g

(k = 0 means the reference job)

$$f_{ijg} = \begin{cases} 1 \text{ if job } j \text{ can be processed on machine } i \text{ of stage } g \\ 0 \text{ otherwise} \end{cases}$$

$$h_{jg} = \begin{cases} 1 \text{ if job } j \text{ has an operation in stage } g \\ 0 \text{ otherwise} \end{cases}$$

4.4 Mathematical Model

$$Min\ Z = \sum_{j=1}^{n}\sum_{i=1}^{m_G} w_j . t_{ijG}$$

Subject to:

$$\sum_{i=1}^{m_g} x_{ijg} = 1, \qquad j = 1, \ldots, n;\ g = 1, \ldots, G \qquad (1)$$

$$c_{ijg} \geq x_{ijg}\left(a_{ig} + r_{ijg}\right) + y_{i0jg} . s_{i0jg},\ \ j = 1, \ldots, n;\ \ g = 1, \ldots, G;\ \ i = 1, \ldots, m_g \qquad (2)$$

$$c_{ij1} \geq x_{ij1}\left(e_{ij1} + r_{ij1}\right), \qquad j = 1, \ldots, n;\ \ i = 1, \ldots, m_1 \qquad (3)$$

$$c_{ijg} \leq M x_{ijg}, \qquad j = 1, \ldots, n;\ \ g = 1, \ldots, G;\ i = 1, \ldots, m_g \qquad (4)$$

$$CT_{jg} \geq \underset{i}{max}\left(c_{ijg}\right),\ \ j = 1, \ldots, n;\ \ g = 1, \ldots, G;\ i = 1, \ldots, m_g \qquad (5)$$

$$\begin{aligned} c_{ijg} \geq CT_{j(g-1)} + h_{jg} . (y_{ikjg} . s_{ikjg} + x_{ijg} . r_{ijg}), \\ k = 1, \ldots, n;\ j = 1, \ldots, n;\ \ g = 2, \ldots, G;\ i = 1, \ldots, m_g \end{aligned} \qquad (6)$$

$$\begin{aligned} c_{ijg} - c_{ikg} + M\left(1 - y_{ikjg}\right) \geq x_{ijg}\left(r_{ijg} + s_{ikjp}\right), \\ k = 1, \ldots, n;\ j = 1, \ldots, n;\ \ g = 1, \ldots, G;\ i = 1, \ldots, m_g;\ k < j \end{aligned} \qquad (7)$$

$$\begin{aligned} c_{ikg} - c_{ijg} + M\left(1 - y_{ijkg}\right) \geq x_{ikg}\left(r_{ikg} + s_{ijkp}\right), \\ k = 1, \ldots, n;\ j = 1, \ldots, n;\ \ g = 1, \ldots, G;\ i = 1, \ldots, m_g;\ k < j \end{aligned} \qquad (8)$$

$$\sum_{\substack{k=0 \\ and \\ k\neq j}}^{n} y_{ikjg} = x_{ijg}\,, \qquad j = 1, \dots, n;\ g = 1, \dots, G;\ i = 1, \dots, m_g \tag{9}$$

$$\sum_{\substack{j=0 \\ and \\ j\neq k}}^{n} y_{ikjg} \leq 1\,, \qquad k = 0, \dots, n;\ g = 1, \dots, G;\ i = 1, \dots, m_g \tag{10}$$

$$y_{ikjg} \leq x_{ikg}, \qquad j = 1, \dots, n; k = 1, \dots, n; g = 1, \dots, G; i = 1, \dots, m_g;\ j \neq k \tag{11}$$

$$t_{ijG} \geq c_{ijG} - d_j, \qquad i = 1, \dots, m_G;\ j = 1, \dots, n \tag{12}$$

$$t_{ijG} \geq 0, \qquad i = 1, \dots, m_G;\ j = 1, \dots, n \tag{13}$$

4.5 Model Description

The proposed mathematical model is a mixed (binary) integer-linear programming model that entails both real and binary (0/1) integers. The objective function of the mathematical model focuses on minimizing the weighted tardiness of all jobs released in the current planning horizon. Constraint (1) states that each job should be processed only on one machine in a given stage. Constraints (2) ensures that the completion time a job on a given machine is at least equal to or greater than the machine’s availability and runtime of that job on the machine, after accounting for the sequence dependent set up time. Constraint (3) also ensures that the completion time of a job on a given machine in stage 1 is always equal to or greater than the release time of the job in stage 1 and the runtime of the job in that stage. Constraint (4) states that the completion time of a job on a machine in a given stage is zero, if its operation is not performed on that machine. Constraints (5) and (6) jointly ensure that the completion time of a job in a given stage is greater than the completion time of the job in previous stage plus the runtime and the sequence-dependent setup time of the job in that stage. Constraints (7) and (8) together ensure that two jobs are not processed at the same time on a machine in a given stage. Constraint (9) states that a job must be preceded by only one job if it has an operation on a particular machine in a given stage. Constraint (10) guarantees that a job can only transfer to at most one job if it has an operation on a particular machine in a given stage whereas constraint (11) ensures that a job cannot transfer to another job if it is not scheduled to be processed on a particular machine of a given stage. Constraint (12) ensures that tardiness of a job is greater than or equal to the difference between its completion time in the last stage and its due date. Finally, constraint (13) guarantees that only positive values for tardiness are considered.

4.6 Computational Complexity of the Research Problem

Research problems, in the field of optimization, can be characterized in a comprehensive manner by using mathematical models. Recent advancements in technology have greatly enhanced the processing speed of computer and have dramatically increased the speed of optimization solvers. Despite these technological advancements, often only small-sized problems can be efficiently solved to optimality by using optimization software.

The computational complexity for this research problem can be illustrated by considering a special case in total weighted tardiness problem. Lenstra et al. (1977) proved that scheduling jobs with equal weights on a single-machine is NP-hard in the strong sense. Note that scheduling jobs on a single machine with equal weights is a special case of unrelated parallel machines with variable weights of jobs. Furthermore, scheduling on unrelated parallel machines is a special case of flexible flow shop scheduling. We therefore conclude that if the special case of the research problem is strongly NP-hard then the original research problem must also be NP-hard in the strong sense.

Obtaining a polynomial time solution (by using optimization software) for a NP-hard problem is very unlikely. As mentioned earlier, only small-sized problems can be efficiently solved by using optimization software. Other implicit enumeration methods such as branch and bound technique can be used to solve small problem instances in a reasonable computational time. But for medium and large problem instances, the branch and bound technique will often fail to identify the optimal solution. Even if it identifies the optimal solution, the computational time required to solve these problem instances can be extremely large. Therefore, we need to develop a solution algorithm for optimally or near-optimally solving the medium and small problem instances in an efficient manner.

In the past, researchers have utilized tabu search-based heuristics to find optimal/near optimal solutions for industry-size problems. Barnes et al. (1995) gave an overview of the application of tabu search heuristics on problems involving single machine, parallel machines, flow shop and job shop. The use of tabu search-based heuristics has been successfully demonstrated by Glass et al. (1994), Suresh and Chaudhary (1996), and Logendran and Subur (2000) in their research work.

5 HEURISTIC ALGORITHM

5.1 Introduction

The heuristic algorithm employs Tabu Search, which was first proposed by Glover (1986), as the mechanism to explore the solution space. The search space of a tabu search is essentially all the solutions (feasible and infeasible) that can be considered during the search. Over the past few years, several researchers have utilized tabu search as a mechanism to solve large combinatorial optimization problems in practical settings. Researchers have used Tabu search heuristics to solve a wide range of problems including the traveling salesman problems, scheduling problems, product delivery and routing problems, and manufacturing cell design problems. It has been proven that tabu search is capable of providing optimal or near optimal solution for complex combinatorial problems in an effective manner. Tabu search is superior to other search mechanisms primarily because it has the capability of overcoming the limitations of local optimality. Tabu search can steer the search mechanism from one solution state to another by tactically constraining and releasing the attributes of the search process. This is achievable because tabu search employs flexible memory functions that record search information of varying time spans. Intensification and diversification of the search to obtain a better solution can be attained by employing long-term memory with minimum and maximum frequencies.

Further details on tabu search including the genesis, fundamental concepts, advanced setting and guiding principle can be found in Glover (1989, 1990a and 1990b). A detailed description of the tabu search mechanism has been presented in the next section. Thereafter, five different mechanisms to obtain the initial solution of the research problem are presented. The steps associated with the generation of neighborhood have also been presented, followed by an example problem that illustrates the application of the heuristic algorithm.

5.2 Tabu Search Mechanism

Tabu search mechanism is primarily built on three features (Glover, 1990b):

1. Tabu search mechanism effectively employs the flexible memory structures to store information during the search process. This feature allows the evaluation criteria and historical search information to be exploited more comprehensively than by rigid memory structures (such as branch-and-bound mechanism) or by memoryless systems (such as simulated annealing and other randomized approaches).
2. A control mechanism that is directed towards the interplay between restricting and releasing the constraints during the search process (embodied in the tabu restrictions and aspiration criteria).
3. The combination of memory functions of different time spans, from short term to long term, to implement strategies for intensifying and diversifying the search.

Tabu search is a refined form of a popular heuristic, namely the hill-climbing heuristic. The hill-climbing heuristic starts with an initial solution and thereafter, depending upon whether the primary objective is minimization or maximization, moves progressively in a unidirectional path for identifying a better solution. Since the hill-climbing heuristic always searches for a better solution, the search terminates whenever a local optimum is found, well before exploring the entire search space. In contrast, the tabu search overcomes the inherent difficultly of being trapped in the local optimum by settling for a solution that is inferior to the previous solution.

Tabu search requires an initial solution to begin the search. Therefore the initial solution can be considered as a matchstick to initiate the search. The initial solution can be generated either arbitrarily or systematically. Additionally, the initial solution can either be feasible or infeasible. Logendran and Subur (2000) have shown that a quality initial solution can lead to the optimal/near-optimal solution much more efficiently. If the initial solution is infeasible then the search space is much wider and therefore it takes a much longer time to get to an optimal/near optimal solution. On the contrary, a better-quality initial solution narrows down the search space and hence it speeds up the process of identifying the optimal/near optimal solution. We propose five different initial solution finding mechanisms that serve as a starting point. These initial solution finding mechanisms have been developed methodically to ensure that the search algorithm performs as efficiently as possible. All five initial solution finding mechanisms are explained in detail in the next section.

By perturbing the initial solution, we can go about finding alternate solutions in the nearest neighborhood. The neighborhood solutions are evaluated by a performance criterion, which in this research is the total weighted tardiness. Each neighborhood solution has to pass a tabu-status check in order to be considered as a permissible move. The objective of the tabu restriction is to take the search process beyond the points of local optimality while ensuring high quality moves at each step. The tabu restrictions are stored in a list known as tabu list. The tabu list entails the recently applied moves that are necessary to move the search from one solution state to another. The moves are recorded exactly in the order in which they are made. A tabu-list has an associated size which determines the amount of time a particular tabu move is enforced. The size of the tabu-list predominantly depends upon size of problems being investigated. This necessitates conducting a thorough experimentation to determine an appropriate size of the tabu-list for a given research problem.

The search process is prevented from revisiting the earlier found solution by restricting the search to moves that are not tabu. In certain instances, a tabu move may result in a solution that is superior to the one found so far. Hence, an aspiration criterion is used to offset tabu restrictions. An aspiration criterion is used as a condition that a tabu move has to satisfy in order to be released from tabu restriction. For a more detailed description please refer to Chapter 5.5. If a tabu move results in a solution that is better than all previously found solution, the tabu restriction can be overridden. Thus an aspiration criterion gives a tabu move a second opportunity to be considered during the search process. After testing all neighborhood solutions against the tabu status and the aspiration criterion, the move that results in the best solution is chosen for future perturbation. This move is stored in the candidate list (CL). Before admitting a solution into the CL, the solution is checked against all other entries of the CL in order to avoid a possible duplication. The entire process is repeated until a criterion to terminate the search is satisfied.

A search can be concluded by employing one of the several terminating conditions. One way to end the search is to set a limit to the size of the IL. In other words, the search will terminate when a certain size of the CL is achieved. Another method is to terminate the search is to let the entire search process run until a certain number of consecutive iterations that do not result in an improvement. Limiting the computational time is yet another way of terminating the search process.

The tabu-list discussed above is the short-term memory component of the tabu search. Besides employing short-term memory to execute the search, we can also utilize the long-term memory function. The long-term memory can be used to direct the search in regions that have been historically found good (intensification process) or in regions that were not thoroughly explored previously (diversification process). Relevant information regarding the long-term memory is stored in a frequency matrix that keeps track of all previous moves.

5.3 Initial Solution

The initial solution finding mechanism serves as a matchstick to initiate the search. In order for the search to begin, the search algorithm must be provided with an initial solution. In the past, researchers have used simple dispatching rules to solve problems aimed at minimizing the tardiness. These problems include minimizing the weighted tardiness, total tardiness and maximum tardiness. Some of the simple dispatching rules used to solve tardiness related problems include Earliest Due Date (EDD), Shortest Processing Time (SPT), Minimum Slack (MSLACK) and Slack per Remaining Processing Time (S/RPT). While the EDD and SPT dispatching rules are time-independent (i.e. the job priority depends upon the job and machine data and it remains the same throughout the scheduling horizon), the MSLACK and S/RPT dispatching rules are time-dependent (i.e. job priority depends upon the time when machines become available after processing a preceding job).

Simple dispatching rules only use a single attribute to attain its objective. An attribute can be defined as a property that belongs to a job or the machine environment under consideration such as the job processing time, job due date, job release time, or job waiting time. Simple dispatching rules are not readily used in an industrial setting where more than one attribute determines a good schedule. Industrial practice often benefits from the use of composite dispatching rules to obtain a good schedule. Unlike a simple dispatching rule, a composite dispatching rule incorporates several job and machine attributes. A composite dispatching rule is a function made up of various attributes and some scaling parameters.

In the past, researchers have proposed several composite dispatching rules for various machine environments. Vepsalainen and Morton (1987) proposed a composite dispatching rule called apparent tardiness heuristic (ATC) to solve weighted tardiness problem on a single machine under the assumption of static job release and static machine availability time. ATC is an amalgamation of two simple dispatching heuristics, namely, the weighted shortest processing time (WSPT) and the minimum slack (MS). The WSPT is a variation of another simple dispatching heuristic called the shortest processing time (SPT). SPT heuristic is used to minimize the mean flow time. Therefore WSPT heuristic is used to minimize the weighted mean flow time. The MS heuristic, on the other hand, maximizes the minimum lateness in a single machine environment.

Lee et al. (1997) extended the ATC rule and suggested another composite dispatching heuristic called Apparent Tardiness Cost with Setup (ATCS). The ATCS heuristic incorporates sequence dependent setup times for solving problems related to weighted tardiness, on a single machine with static job releases and static machine availability. The ATCS rule utilizes three different simple composite dispatching rules namely, the WSPT rule, the MS rule and the Shortest setup time (SST). Few other composite dispatching rules include the Dynamic Composite Rule/DCR (Conway et al., 1967) and the Cost Over Time/COVERT (Carroll, 1965).

Previous research on tabu search (Logendran and Subur, 2004) has shown that a superior quality initial solution may contribute to identifying a better quality final/best solution in an efficient manner. Thus we propose five different methods to generate the initial solution for the tabu search. These five initial solutions are as follows: Earliest Due Date (EDD), Least Flexible Job/Least Flexible Machine (LFJ/LFM), Lowest Weighted Tardiness (LWT), Due Date to Weight Ratio (DDW) and Hybrid Critical Ratio (HCR). The mechanism for finding each of Initial solution will be presented in detail later. We first present the set of notations that will be used throughout the development of the algorithm.

Notations

$i = 1, 2, \ldots, G$ stages

$k = 1, 2, \ldots, f_i$ machines in stage i

$m_{ki} = k^{\text{th}}$ machine of stage i

NSJ = set of unscheduled jobs

p = 1, 2,...,l (total number of projects)

j = 1, 2,…,n_p number of jobs in project p

J_{jp} = job j of project p (j = 1, 2,…,n_p; p = 1,2,…,l)

Parameters

r_{jpki} = run time of job j of project p on k^{th} machine of stage i;

w_{jp}= weight assigned to job j of project p;

e_{jp} = release time of job j of project p;

d_{jp} = due date of job j of project p;

$s_{(j'p')(jp)ki}$ = setup time required to change over from job j' of project p' to job j of project p on k^{th} machine of stage i (where p=p' is allowed since the changeover of jobs can occur within the same project);

a_{ki} = initial availability time of k^{th} machine of stage i;

td_{jpki} = tardiness of job job j of project p on k^{th} machine of stage i;

tm_{k*1} = release time of k^{th} machine of stage 1 (after processing a job)

Variables

$CT(J_{jp}, m_{ki})$ = Completion time of job j of project p on k^{th} machine of stage i

$ST(J_{jp}, m_{ki})$ = Start time of job j of project p on k^{th} machine of stage i

5.3.1 Earliest Due Date (EDD)

IS1 aims at giving the highest priority to the job that has the earliest due date (EDD) among the jobs that can be processed on an available machine. If two jobs have the same due date, ties are broken in favor of the job that has more weight. If two jobs have the same EDD and weight, then ties are broken in favor of the job that has the smallest project identification number. If two jobs have the same EDD, weight and same project identification number, ties are broken in favor of the job that has the smallest job id. The machines are selected in the order of increasing available time (a_i). Should there be a tie among two or more machines, the machine that has the lowest machine identification number is selected. The following steps give a comprehensive illustration of the methodology associated in the development of the initial solution based on the EDD rule.

1. Initially, set $t = 0$ and $tm_{k1} = \mathrm{a}_{k1}$ $\forall$ $k \in f_1$. Include all jobs in *NSJ*.
2. Select the machine/unit (k) that has the minimum tm_{k1}. If there is more than one machine with minimum tm_{k1}, break ties by choosing the machine with the smallest machine identification number. Let the selected machine be k^*. Set $t = tm_{k^*1}$.
3. Let *SJ* = the set of job/jobs (belonging to various projects) released at or earlier than t, and that can be processed on k^* ($SJ \subset NSJ$).
    a. If $SJ = \phi$, find a job/jobs from *NSJ* that can be processed on k^* and have the minimum e_{jp}.
        i. If none of the jobs in NSJ can be processed on k^*, exclude k^* from future consideration. Go to step 6.
        ii. If only one job is found, select this job and assign it to k^*. Go to step 4.
        iii. If two or more jobs are found, break ties in favor of job that has the earliest due date, followed by highest weight. If the job ties still exist, break ties in favor job that has the smallest project identification number followed by smallest job identification number. Assign the selected job to k^*. Go to step 4.
    b. If *SJ* has only one job, assign the job to k^*. Go to step 4.

c. If *SJ* has two or more jobs, break ties in favor of job that has the earliest due date followed by highest weight. If the job ties still exist, break ties in favor of job that has the smallest project identification number followed by smallest job identification number. Assign the selected job to k^*. Go to step 4.

4. Let $J_{j^*p^*}$ = the selected job; evaluate the start time and completion time of job using the following equations:

a. If $e_{j^*p^*}$ is less than or equal to tm_{k^*1}, then set ST $(\mathrm{J}_{j^*p^*},\mathrm{m}_{k^*1}) = tm_{k^*1}$ and $CT(J_{j^*p^*},m_{k^*1}) = ST\ (J_{j^*p^*},m_{k^*1}) + s_{(j'p')\ (j^*p^*)k^*1} + r_{j^*\,p^*k^*1}$.

b. If $e_{j^*p^*}$ is greater than tm_{k^*1}, then set $ST\ (J_{j^*p^*},m_{k^*1}) = e_{j^*p^*}$. Next evaluate $(e_{j^*p^*} - tm_{k^*1})$.

i. If $(e_{j^*p^*} - tm_{k^*1})$ is greater than or equal to $s_{(j'p')\ (j^*p^*)(k^*1)}$, then $CT(J_{j^*p^*},m_{k^*1}) = ST\ (J_{j^*p^*},m_{k^*1}) + r_{j^*\,p^*k^*1}$.

ii. If $(e_{j^*p^*} - tm_{k^*1})$ is less than $s_{(j'p')\ (j^*p^*)(k^*1)}$ $CT(J_{j^*p^*},m_{k^*1}) = ST\ (J_{j^*p^*},m_{k^*1}) + [s_{(j'p')\ (j^*p^*)k^*1} - (e_{j^*p^*} - \mathrm{tm}_{k^*1})] + r_{j^*\,p^*k^*1}$.

5. Set $tm_{k^*1} = CT(J_{j^*p^*},m_{k^*1})$. Eliminate $J_{j^*p^*}$ from *NSJ*.
6. If $NSJ \neq \phi$, go to step 2.

The algorithmic procedure in step 4 needs further explanation. The equations in step 4 illustrate the evaluation of completion times of jobs in stage 1. Since this research problem involves sequence-dependent setup times, we primarily investigate two cases for evaluating the completion times of jobs (Step 4 (a) and (b)). The first case, (Step 4 (a), assumes that a job is released before a machine is available. In this case, the setup can be started only when the machine becomes available. The second case, Step 4 (b), pertains to the instance when the machine becomes available before the job. In this case, we can perform an anticipatory setup, starting when the machine becomes available, and can complete either the entire setup (Step 4 (b)(i)) or partial setup for the job (Step 4 (b)(ii)).

5.3.2 Least Flexible Job and Least Flexible Machine (LFJ/LFM)

Centeno and Armacost (1997) proposed an algorithm for scheduling jobs on parallel machines with dynamic job release time, due dates and different machine capabilities. The purpose of their research was to minimize the maximum lateness. Lateness of a job is evaluated as the completion time minus the due date. Thus Lateness of a job can either be a negative, zero or a positive value. Lateness of a job should not be confused with the tardiness of a job. Tardiness of a job can only be zero or positive value. Tardiness of a job can never take a negative value.

LFJ (least flexible job) can be defined as the job that can be processed on least number of machines and LFM (least flexible machine), which can be defined as the machine that can process the least number of jobs. The LFJ/LFM approach to IS2 gives priority to those jobs that are least flexible and machines that are least capable. If two or more jobs are available at the same time, ties are broken in favor of the job that is least flexible. If two or more machines are available at the same time, ties are broken in favor of the machine that is least flexible. Other tie braking rules for IS2 are similar to that of IS1.

The LFJ rule prevents jobs from being late due to their inflexibility. The LFM rule ensures that less capable machines get a fair share of job assignment in comparison to more capable machines. The following steps give a comprehensive illustration of the methodology associated in the development of the initial solution based on the LFJ/LFM rule.

1. Initially, set $t = 0$ and $tm_{k1} = \mathrm{a}_{k1}$ $\forall$ $k \in f_1$. Include all jobs in *NSJ*.
2. Check if any $e_{jp} \leq t$. If yes, then go to step 4.
3. Set $t = \min\, [e_{jp}]$ where $J_{jp} \in NSJ$.
4. Choose the least flexible job with $e_{jp} \leq t$. If two or more jobs are chosen, select the job with minimum e_{jp}. If the job ties still exist, break ties in favor of job that has

the smallest project identification number followed by smallest job identification number.

5. Let $J_{j^*p^*}$ = the selected job. Find the least flexible machine that can process $J_{j^*p^*}$ and is currently idle. If two or more machines are found, break ties in favor of machine with the smallest machine index.
    - a. If all the machines that are capable of processing $J_{j^*p^*}$ are busy, go to step 8.
    - b. Let the selected machine be k^*. Go to step 6.
6. Evaluate the start time and completion time of job using the following equations:
    - a. If $e_{j^*p^*}$ is less than or equal to tm_{k^*1}, then set $ST\ (J_{j^*p^*}, m_{k^*1}) = tm_{k^*1}$ and $CT(J_{j^*p^*}, m_{k^*1}) = ST\ (J_{j^*p^*}, m_{k^*1}) + s_{(j'p')\ (j^*p^*)k^*1} + r_{j^*\ p^*k^*1}$.
    - b. If $e_{j^*p^*}$ is greater than tm_{k^*1}, then set $ST\ (J_{j^*p^*}, m_{k^*1}) = e_{j^*p^*}$. Calculate $(e_{j^*p^*} - tm_{k^*1})$.
        - i. If $(e_{j^*p^*} - tm_{k^*1})$ is greater than or equal to $s_{(j'p')\ (j^*p^*)(k^*1)}$, then $CT(J_{j^*p^*}, m_{k^*1}) = ST\ (J_{j^*p^*}, m_{k^*1}) + r_{j^*\ p^*k^*1}$.
        - ii. If $(e_{j^*p^*} - tm_{k^*1})$ is less than $s_{(j'p')\ (j^*p^*)(k^*1)}$, then $CT(J_{j^*p^*}, m_{k^*1}) = ST\ (J_{j^*p^*}, m_{k^*1}) + [s_{(j'p')\ (j^*p^*)k^*1} - (e_{j^*p^*} - tm_{k^*1})] + r_{j^*\ p^*k^*1}$.
7. Set $tm_{k^*1} = CT(J_{j^*p^*}, m_{k^*1}) = t$. Eliminate $J_{j^*p^*}$ from NSJ. Go to step 9.
8. Set $t = \min[tm_{k1}]\ \forall\ k \in f_1$ and k can process $J_{j^*p^*}$; go to step 2.
9. If $NSJ \neq \phi$, go to step 2; otherwise, stop.

5.3.3 Lowest Weighted tardiness (LWT)

For IS3, we use LWT (lowest weighted tardiness) mechanism to commence the search. Tardiness for job *j* of project *p* on k^{th} machine of stage *i* (td_{jpki}) can be evaluated as max [0, CT (J_{jp}, m_{ik}) - d_{jp}] and weighted tardiness can be evaluated as (td_{jpki}) w_{jp}. The tie braking rules for machines and jobs are similar to IS1. If two jobs have the same LWT, ties are broken in favor of the job that has the smallest project identification number. If two or more jobs have same LWT and same project identification number then the ties are broken in favor of the job with the smallest job id. The machines are selected in the order of increasing available time (a_i). Should there be a tie among two or more machines, the machine that has the lowest machine identification number is selected.

It should be noted that to find the completion time of jobs, we need a schedule that would tell us the sequence of jobs on each machine in every stage. We propose a mechanism based on C_{max} (the maximum completion time (makespan) of all jobs released) to come up with a provisional schedule. Although the real C_{max} is schedule-dependent, a thorough estimation scheme has to be developed for accurately estimating C_{max} that is schedule-independent. Given the job release time, job setup time, job runtime and machine availability time, we propose the following equation to estimate the C_{max} for the i^{th} stage:

$$C_{\mathrm{max,i}} = \frac{\sum_{p=1}^{l} \sum_{j=1}^{n_p} \left[\frac{\sum_{\substack{k=1 \\ and \\ r_{jpki>0}}}^{f_i} \max\left[\mathrm{e}_{jpi},\ a_{ki} + (\beta \times \bar{s}_{jpki})\right] + r_{jpki}}{m_j} \right]}{f_i} \text{ where } i = 1,2, \dots ,17$$

where m_j is the total number of machines on which job *j* can be processed in stage i, $\bar{s}_{jpki}$ is the average setup time of job *j* of project *p* on k^{th} machine of stage *i* and β is introduced as an adjustment to the average setup time. The need for incorporating β originates from the fact that in reality, a quality schedule would try to changeover from one job to another that requires the smallest setup time on a particular machine. Hence considering average setup time for jobs in estimating C_{max} disregarding β, would not provide an accurate estimate of the makespan for the first stage. We further define coefficient of variation (CV) for the sequence-dependent setup times for a job on a machine as CV = $s/\bar{x}$ where s is the sample standard deviation and $\bar{x}$ is the mean. In case of setup times being sequence independent, the standard deviation of the data points (i.e., the sequence-dependent setup times of a job on a machine) will be equal to zero, forcing CV to be equal to zero. We surmise that a linear relationship holds true between β and CV, and suggest the following set of end points for the purpose of interpolation: CV = 0.01 corresponds to β =.9, and CV = 1.0 corresponds to β = 0.1.

The average completion time of a job evaluated for the first stage serves as the release time for jobs in the second (following) stage. We propose the following equation to estimate the release time of a job in $(i+1)^{th}$ stage:

$$e_{jp(i+1)} = \frac{\sum_{\substack{k=1 \\ and \\ r_{jpki>0}}}^{f_i} \max\left[\mathrm{e}_{jpi},\ a_{ki} + (\beta \times \bar{s}_{jpki})\right] + r_{jpki}}{m_j} \text{ where } i = 1,2, \dots 16$$

To estimate a reasonable completion time for a particular job in a specific stage i (or equivalently the release time of a job in $(i+1)^{th}$ stage), we compare the release time of a job in stage i with the machine's availability time combined with the average setup time of that job on that machine. The rationale to go after the larger of the two evaluated numbers originates from the fact that if a job is released before a machine is available then the job has to wait until the machine is made available and the setup is performed on the machine or vice-versa. Recall that not all of the jobs have operation in each and every stage. If such were the case for one or more jobs, the evaluated completion time of the job serves as the release time for the next immediate stage, where the job has an operation. Notice that the $C_{\max}$ increases progressively through stages 1-17. Therefore, the following condition holds true:

$$C_{\max,i+1} \geq C_{\max,i} \ (\forall\ i = 1,2, \dots 16)$$

When all jobs choose to skip a particular stage, $C_{\max,i+1} = C_{\max,i}$; for all other stages $C_{\max,i+1} > C_{\max,i}$.We evaluate the $C_{\max}$ for stages 1 through 17 and use $C_{\max}$ evaluated for stage 17 to estimate the maximum completion time of all jobs for a given problem. The scheduling steps that use the LWT dispatching rule can be documented as follows:

1 Initially, set $t = 0$ and $tm_{k1} = a_{k1}\ \forall\ k \in f_1$. Include all jobs in *NSJ*.

2 Select the machine/unit (k) that has the minimum tm_{k1}. If there is more than one machine with minimum tm_{k1}, break ties by choosing the machine with the smallest machine identification number. Let the selected machine be k^*. Set $t = tm_{k^*1}$.

3 Let *SJ* = the set of job/jobs (belonging to various projects) released at or earlier than t, and that can be processed on k^* ($SJ \subset NSJ$).

   a. If $SJ = \phi$, find a job/jobs from *NSJ* that can be processed on k^* and have the minimum e_{jp}.

      i. If none of the jobs in NSJ can be processed on k^*, exclude k^* from future consideration. Go to step 6.

      ii. If only one job is found, select this job and assign it to k^*. Go to step 4.

      iii. If two or more jobs are found, break ties in favor of job that has the lowest weighted tardiness. If the job ties still exist, break ties in favor of job that has the smallest project identification number followed by smallest job identification number. Assign the selected job to k^*. Go to step 4.

   b. If *SJ* has only one job, assign the job to k^*. Go to step 4.

   c. If *SJ* has two or more jobs, break ties in favor of job that has the lowest weighted tardiness. If the job ties still exist, break ties in favor of job that has the smallest project identification number followed by the smallest job identification number. Assign the selected job to *k**. Go to step 4.

4 Let J_{j*p*} = the selected job; Evaluate the start time and completion time of job using the following equations:

   a. If e_{j*p*} is less than or equal to tm_{k*1}, then set $ST\ (J_{j*p*}, m_{k*1}) = tm_{k*1}$ and $CT(J_{j*p*}, m_{k*1}) = ST\ (J_{j*p*}, m_{k*1}) + s_{(j'p')\ (j*p*)k*1} + r_{j*\ p*k*1}$.

   b. If e_{j*p*} is greater than tm_{k*1}, then set $ST\ (J_{j*p*}, m_{k*1}) = e_{j*p*}$. Next evaluate $(e_{j*p*} - tm_{k*1})$.

      i. If $(e_{j*p*} - tm_{k*1})$ is greater than or equal to $s_{(j'p')\ \ (j*p*)(k*1)}$,then $CT(J_{j*p*}, m_{\ k*1}) = ST\ (J_{j*p*}, m_{\ k*1}) + r_{j*\ p*k*1}$.

      ii. If $(e_{j*p*} - tm_{k*1})$ is less than $s_{(j'p')\ (j*p*)(k*1)}$, then $CT(J_{j*p*}, m_{k*1}) = ST\ (J_{j*p*}, m_{k*1}) + [s_{(j'p')\ (j*p*)k*1} - (e_{j*p*} - \mathrm{tm}_{k*1})] + r_{j*\ p*k*1}$.

5 Set $tm_{k*1} = CT(J_{j*p*}, m_{k*1})$. Eliminate J_{j*p*} from *NSJ*.

6 If $NSJ \neq \phi$, go to step 2.

5.3.4 Due Date Weight Ratio (DDW)

IS4 mechanism initiates jobs to be processed using the due date to weight (DDW) ratio. DDW can be defined as d_{jp}/w_{jp}. The motive is to expand the ideas presented in IS1. IS1 is myopic to the weights assigned to the various jobs released during the planning horizon. In IS1, we only dealt with the EDD, (though we took weight into consideration for breaking ties) but in IS4 we attempt to combine the EDD factor and the weight assigned to a job. The logic behind DDW ratio is that a job that has the earliest due date and most weight will have the least DDW and therefore should be given a higher priority over other jobs.

The tie braking rules for machines and jobs are analogous to IS1. IS4 begins with prioritizing jobs with the lowest DDW ratio. If two jobs have the same DDW ratio, ties are broken in favor of the job that has the smallest project identification number. If two or more jobs have same DDW ratio and same project identification number then the ties are broken in favor of the job with smallest job id. The machines are selected in the order of increasing available time (a_i). Should there be a tie among two or more machines, the machine that has the lowest machine identification number is selected.

A method to generate the initial solution is developed based on DDW ratio. The steps associated with this method can be documented as follows:

1. Initially, set $t = 0$ and $tm_{k1} = \mathrm{a}_{k1}\ \forall\ k \in f_1$. Include all jobs in *NSJ*.
2. Select the machine/unit (k) that has the minimum tm_{k1}. If there is more than one machine with minimum tm_{k1}, break ties by choosing the machine with the smallest machine identification number. Let the selected machine be k*. Set $t = tm_{k*1}$.
3. Let *SJ* = the set of job/jobs (belonging to various projects) released at or earlier than t, and that can be processed on k* ($SJ \subset NSJ$).
   a. If $SJ = \phi$, find a job/jobs from *NSJ* that can be processed on k* and have the minimum e_{jp}.

i. If none of the jobs in NSJ can be processed on k^*, exclude k^* from future consideration. Go to step 6.

ii. If only one job is found, select this job and assign it to k^*. Go to step 4.

iii. If two or more jobs are found, break ties in favor of job that has the smallest due date to weight ratio (DDW). If the job ties still exist, break ties in favor of job that has the smallest project identification number followed by smallest job identification number. Assign the selected job to k^*. Go to step 4.

b. If *SJ* has only one job, assign the job to k^*. Go to step 4.

c. If *SJ* has two or more jobs, break ties in favor of job that has the smallest due date to weight ratio (DDW). If the job ties still exist, break ties in favor of job that has the smallest project identification number followed by smallest job identification number. Assign the selected job to k^*. Go to step 4.

4. Let $J_{j^*p^*}$ = the selected job; Evaluate the start time and completion time of job using the following equations:

   a. If $e_{j^*p^*}$ is less than or equal to tm_{k^*1}, then set $ST\ (J_{j^*p^*}, m_{k^*1}) = tm_{k^*1}$ and $CT(J_{j^*p^*}, m_{k^*1}) = ST\ (J_{j^*p^*}, m_{k^*1}) + s_{(j'p')\ (j^*p^*)k^*1} + r_{j^*\,p^*k^*1}$.

   b. If $e_{j^*p^*}$ is greater than tm_{k^*1}, then set $ST\ (J_{j^*p^*}, m_{k^*1}) = e_{j^*p^*}$. Next evaluate $(e_{j^*p^*} - tm_{k^*1})$.

      i. If $(e_{j^*p^*} - tm_{k^*1})$ is greater than or equal to $s_{(j'p')\ (j^*p^*)(k^*1)}$, then $CT(J_{j^*p^*}, m_{\ k^*1}) = ST\ (J_{j^*p^*}, m_{\ k^*1}) + r_{j^*\,p^*k^*1}$.

      ii. If $(e_{j^*p^*} - tm_{k^*1})$ is less than $s_{(j'p')\ (j^*p^*)(k^*1)}$ $CT(J_{j^*p^*}, m_{k^*1}) = ST\ (J_{j^*p^*}, m_{k^*1}) + [s_{(j'p')\ (j^*p^*)k^*1} - (e_{j^*p^*} - \mathrm{tm}_{k^*1})] + r_{j^*\,p^*k^*1}$.

5. Set $tm_{k^*1} = CT(J_{j^*p^*}, m_{k^*1})$. Eliminate $J_{j^*p^*}$ from *NSJ*.

6. If $NSJ \neq \phi$, go to step 2.

5.3.5 Hybrid Critical Ratio (HCR)

The critical ratio, defined as due date/processing time, has been used for due date related objectives in the scheduling literature for a long time. Logendran et. al. (2007) suggested a modified version of critical ratio (known as hybrid critical ratio or HCR) for problems that involved sequence dependent setup times. Taking advantage of their work, we propose IS5 (which is a refined version of IS4). In IS5, we include the sequence-dependent setup time and run time of a job besides due date and weight. The hybrid critical ratio is given by $\{d_{jp}/[w_{jp}*(s_{(j'p')(jp)ki} + r_{jpki})]\}$. The job that has the smallest HCR is given preference. The logic behind HCR calculation is similar to the DDW ratio. The only difference is that we introduce two new variables in the denominator.

The tie braking rules for jobs are similar to IS1 but instead of using EDD, we use the HCR. IS5 begins with prioritizing jobs with the lowest HCR ratio. If two jobs have the same HCR ratio, ties are broken in favor of the job that has the smallest project identification number. If two or more jobs have the same HCR ratio and same project identification number, then the ties are broken in favor of the job with smallest job id. The machines are selected in the order of increasing available time (a_i). Should there be a tie among two or more machines, the machine that has the lowest machine identification number is selected.

A method to generate the initial solution is developed based on HCR ratio. The steps associated with this method can be documented as follows:

1. Initially, set $t = 0$ and $tm_{k1} = \mathrm{a}_{k1}\ \forall\ k \in f_1$. Include all jobs in *NSJ*.
2. Select the machine/unit (k) that has the minimum tm_{k1}. If there is more than one machine with minimum tm_{k1}, break ties by choosing the machine with the smallest machine identification number. Let the selected machine be k^*. Set $t = tm_{k^*1}$.
3. Let SJ = the set of job/jobs (belonging to various projects) released at or earlier than t, and that can be processed on k^* ($SJ \subset NSJ$).

a. If $SJ = \phi$, find a job/jobs from *NSJ* that can be processed on k^* and have the minimum e_{jp}.

   i. If none of the jobs in NSJ can be processed on k^*, exclude k^* from future consideration. Go to step 6.

   ii. If only one job is found, select this job and assign it to k^*. Go to step 4.

   iii. If two or more jobs are found, break ties in favor of job that has smallest hybrid critical ratio (HCR). If the job ties still exist, break ties in favor of job that has the smallest project identification number followed by smallest job identification number. Assign the selected job to k^*. Go to step 4.

b. If *SJ* has only one job, assign the job to k^*. Go to step 4.

c. If *SJ* has two or more jobs, break ties in favor of job that has the smallest hybrid critical ratio (HCR). If the job ties still exist, break ties in favor of job that has the smallest project identification number followed by smallest job identification number. Assign the selected job to k^*. Go to step 4.

4. Let $J_{j^*p^*}$ = the selected job; Evaluate the start time and completion time of job using following equations:

   a. If $e_{j^*p^*}$ is less than or equal to tm_{k^*1}, then set $ST\ (J_{j^*p^*}, m_{k^*1}) = tm_{k^*1}$ and $CT(J_{j^*p^*}, m_{k^*1}) = ST\ (J_{j^*p^*}, m_{k^*1}) + s_{(j'p')\ (j^*p^*)k^*1} + r_{j^*\ p^*k^*1.}$

   b. If $e_{j^*p^*}$ is greater than tm_{k^*1}, then set $ST\ (J_{j^*p^*}, m_{k^*1}) = e_{j^*p^*}$. We calculate $(e_{j^*p^*} - tm_{k^*1})$.

      i. If $(e_{j^*p^*} - tm_{k^*1})$ is greater than or equal to $s_{(j'p')\ (j^*p^*)(k^*1)}$, then $CT(J_{j^*p^*}, m_{\ k^*1}) = ST\ (J_{j^*p^*}, m_{\ k^*1}) + r_{j^*\ p^*k^*1}$.

      ii. If $(e_{j^*p^*} - tm_{k^*1})$ is less than $s_{(j'p')\ (j^*p^*)(k^*1)}$ $CT(J_{j^*p^*}, m_{k^*1}) = ST\ (J_{j^*p^*}, m_{k^*1}) + [s_{(j'p')\ (j^*p^*)k^*1} - (e_{j^*p^*} - \mathrm{tm}_{k^*1})] + r_{j^*\ p^*k^*1}$.

5. Set $tm_{k^*1} = CT(J_{j^*p^*}, m_{k^*1})$. Eliminate $J_{j^*p^*}$ from *NSJ*.

6. If $NSJ \neq \phi$, go to step 2.

5.4 Generation of Neighborhood Solutions

The application of tabu search begins with the initial solution as the seed. Two methods are developed to generate a set of neighborhood solutions from a seed. The total weighted tardiness is evaluated for each of the solutions generated by applying these methods. The best solution is then selected as the new seed to generate a new set of neighborhood solutions. This process is repeated aat every iteration of tabu search until the search is terminated. The performance criteria and the steps related to tabu search application are explained in the next section.

In order to generate a set of neighborhood solutions from a chosen seed, two types of move are applied to the seed: Swap moves and insert moves. A swap move is a move that interchanges the positions of two jobs that are assigned to the same machines. An insert move is a move that inserts a job to any machine except the one that it currently occupies. A swap moves allows two jobs from the same or different machines to exchange postions. An insert move allows a job to move from one machine to another.

5.4.1 Swap Move

Let J_{j**p**} and $J_{j''p''}$ be the jobs considered for swap. J_{j**p**} and $J_{j''p''}$ are currently scheduled on machine M_{k*1} and $M_{k'1}$, respectively. Let $[\ldots J_{j*p*}, J_{j**p**}, J_{j***p***}, \ldots]$ be the partial sequence of jobs assigned to M_{k*1} and $[\ldots J_{j'p'}, J_{j''p''}, J_{j'''p'''},\ldots]$ be the partial sequence of jobs assigned to $M_{k'1}$. J_{j**p**} and $J_{j''p''}$ are allowed to exchange positions if *both* the following conditions are satisfied:

1. J_{j**p**} can be processed on $M_{k'1}$ and $J_{j''p''}$ can be processed on M_{k*1}.
2. $e_{j**p**} < CT(J_{j''p''}, M_{k'1})$ and $e_{j''p''} < CT(J_{j**p**}, M_{k*1})$.

If J_{j**p**} and $J_{j''p''}$ satisfied the two conditions, proceed with swapping J_{j**p**} and $J_{j''p''}$. The start time and completion time of J_{j**p**} and $J_{j''p''}$ must be revised. To differentiate the current start and completion times from the revised times, a subscript 'r' is added to the notation such that the revised start time and completion time are denoted by ST_r and CT_r, respectively.

a. If $e_{j^{**}p^{**}}$ is less than or equal to $tm_{k'1}$, then set STr ($J_{j^{*}p^{*}}$,$m_{k'1}$) = $tm_{k'1}$ and $CTr(J_{j^{**}p^{**}},m_{k'1}) = STr\ (J_{j^{**}p^{**}},m_{k'1}) + s_{(j'p')\,(j^{**}p^{**})k'1} + r_{j^{**}\,p^{**}k'1}$.
b. If $e_{j^{**}p^{**}}$ is greater than $tm_{k'1}$, then set $ST\ (J_{j^{**}p^{**}},m_{k'1}) = e_{j^{**}p^{**}}$. Next evaluate ($e_{j^{**}p^{**}} - tm_{k'1}$).
    i. If ($e_{j^{**}p^{**}}$ - $tm_{k'1}$) is greater than or equal to $s_{(j'p')\ (j^{**}p^{**})(k'1)}$, then $CTr(J_{j^{**}p^{**}},m_{\,k'1}) = STr\ (J_{j^{**}p^{**}},m_{\,k'1}) + r_{j^{**}p^{**}k'1}$.
    ii. If ($e_{j^{**}p^{**}}$ - $tm_{k'1}$) is less than $s_{(j'p')\ (j^{**}p^{**})(k'1)}$ $CTr(J_{j^{**}p^{**}},m_{k'1}) = ST\ (J_{j^{**}p^{**}},m_{k'1}) + [s_{(j'p')\,(j^{**}p^{**})(k'1)} - (e_{j^{**}p^{**}} - \mathrm{tm}_{k'1})] + r_{j^{**}\,p^{**}k'1}$.

5.4.2 Insert Move

Let $J_{j^{**}p^{**}}$ is the job considered for insertion on $M_{k'1}$ and $J_{j^{**}p^{**}}$ is currently scheduled on machine $M_{k^{*}1}$. Let [...$J_{j^{*}p^{*}}$, $J_{j^{**}p^{**}}$, $J_{j^{***}p^{***}}$, ...] be the partial sequence of jobs assigned to $M_{k^{*}1}$ and [...$J_{j'p'}$, $J_{j''p''}$, $J_{j'''p'''}$,...] be the partial sequence of jobs assigned to $M_{k'1}$. $J_{j^{**}p^{**}}$ is allowed to be inserted on $M_{k'1}$ if *both* the following conditions are satisfied:

3. $J_{j^{**}p^{**}}$ can be processed on $M_{k'1}$ $e_{j^{**}p^{**}} < CT(J_{j''p''}, M_{k'1})$ an$_{d\ e_{j''p''}<}$ $CT(J_{j^{**}p^{**}}\ _{,}\ M_{k^{*}1})$.

If $J_{j^{**}p^{**}}$ satisfies the two conditions, proceed with inserting $J_{j^{**}p^{**}}$ on $M_{k'1}$. The start time and completion time of $J_{j^{**}p^{**}}$ must be revised. To differentiate the current start and completion times from the revised times, a subscript ‘r’ is added to the notation such that the revised start time and completion time are denoted by ST_r and CT_r, respectively.

c. If $e_{j^{**}p^{**}}$ is less than or equal to $tm_{k'1}$, then set $STr\ (J_{j^*p^*}, m_{k'1}) = tm_{k'1}$ and $CTr(J_{j^{**}p^{**}}, m_{k'1}) = STr\ (J_{j^{**}p^{**}}, m_{k'1}) + s_{(j'p')\ (j^{**}p^{**})k'1} + r_{j^{**}\ p^{**}k'1}$.

d. If $e_{j^{**}p^{**}}$ is greater than $tm_{k'1}$, then set $ST\ (J_{j^{**}p^{**}}, m_{k'1}) = e_{j^{**}p^{**}}$. Next evaluate $(e_{j^{**}p^{**}} - tm_{k'1})$.

   i. If $(e_{j^{**}p^{**}} - tm_{k'1})$ is greater than or equal to $s_{(j'p')\ (j^{**}p^{**})(k'1)}$, then $CTr(J_{j^{**}p^{**}}, m_{k'1}) = STr\ (J_{j^{**}p^{**}}, m_{k'1}) + r_{j^{**}p^{**}k'1}$.

If $(e_{j^{**}p^{**}} - tm_{k'1})$ is less than $s_{(j'p')\ (j^{**}p^{**})(k'1)}$ $CTr(J_{j^{**}p^{**}}, m_{k'1}) = ST\ (J_{j^{**}p^{**}}, m_{k'1}) + [s_{(j'p')\ (j^{**}p^{**})(k'1)} - (e_{j^{**}p^{**}} - \mathrm{tm}_{k'1})] + r_{j^{**}}$

5.5 Steps of Tabu Search

The steps related to the tabu-search mechanism can be documented as follows:

Step 1: Apply swap and insert moves to the initial solution in order to obtain a set of neighborhood solutions. A problem instance with n jobs has $\binom{n}{2} = \frac{n!}{(n-2)!*2!}$ possible combinations of swap moves. Note that the above equation represents any possible combination of two jobs. A swap move, however, is applied to a pair of jobs only if they satisfy the conditions listed in Section 5.4.1.

In applying insert moves, the attempt is to insert jobs to different positions on all the machines that have the capability of processing the job. Insert moves are not attempted on machines that the job is presently occupying. Suppose that a machine currently has k jobs scheduled to be processed. The total number of positions (on that particular machine) to insert a job will be k+1, i.e. all the currently occupied positions plus the last unoccupied position. Note that all the conditions listed in Section 5.4.2 must be satisfied before we can apply the insert move to a job. Applying swap and insert moves provide with a set of solutions considered as the neighborhood solution of the initial solution.

Step 2: Evaluate the total weighted tardiness (TWT) of every solution in the neighborhood, obtained as a result of performing swap and insert moves. As mentioned before, tardiness of a job is evaluated as Max (0, CT_{jp} - d_{jp}) and weighted tardiness is evaluated by multiplying the weight of a job and its tardiness.

Step 3: Select the solution that results in the best (minimum) TWT value. If one or more solutions yield the best TWT, choose the first best solution. Apply the move that results in the best solution to the initial solution. Then update the following parameters used during the search process.

(1) Tabu List: Tabu list consists of the most recent moves (swap or insert). The move that resulted in the best TWT is recorded into the Tabu list. For example, if a swap move resulted in the best TWT then record the pair of jobs swapped into the tabu list. This job pair is not allowed to exchange positions unless an aspiration criterion is satisfied. All such pairs of job that appear in the tabu list (at any given time) indicate that these pairs were swapped at some previous iteration during the search process.

Now suppose that the best solution results after applying an insert move. The tabu list records the job index along with the position and the machine occupied by the job, before the insert move was applied. The job cannot be inserted back at the same position of the same machine unless an aspiration criterion is satisfied. All insert moves recorded in the tabu list indicate that a particular job had been inserted at some position on a machine in some previous iteration.

The entries in the tabu list follow the FIFO (first-in-first-out) rule. This means that the oldest entry is removed and the new one is inserted into the tabu list whenever the tabu list reaches its maximum size. For example, if the size of tabu list is equal to 4 then a move will stay in the tabu list for four iterations. The length of time a move remains tabu depends on the tabu list size. Tabu list size for a particular problem is determined empirically.

Since tabu list stores the most recent moves applied as the search progresses, it is necessary to make the size of tabu list proportional to the total number of possible moves (both swap and insert). The number of possible moves increases as we increase the total number of jobs. Therefore the size of the tabu list is dependent upon the total number of jobs.

Any given problem instance can be categorized as small, medium or large problem. If a given problem has 20 or fewer jobs, it is considered as a small problem instance. If a given problem has 20 jobs or greater but less than or equal to 40, it is considered as a medium problem instance. If a given problem has greater than 40 jobs, it is considered to be a large problem instance. Classification of problem instances (into small, medium or large) was done only after consulting with experts in a manufacturing company where this research can be applied. Based on factual company data and years of experience, the experts helped us to classify the problem into small, medium and large categories.

Two different types of tabu list size are considered in this research: fixed tabu list size and variable tabu list size. Empirical formulae for determining tabu list size were developed after performing detailed experiments for small, medium and large size problems. Instead of fixing tabu list sizes arbitrarily, the idea was to conduct a thorough experimentation to allow best performing values to be input into the search algorithm when problems are solved. For example, to determine the tabu list size for large problem instances, we varied the tabu list size from 1 to a large number (say 30) while holding other relevant parameters (Iterations without Improvement and Entries into the Index List) constant at large values. A particular value that returned the best TWT was noted and we constructed empirical equations based on the experimental results.

Extensive experimentation was performed for each (small medium and large) problem structure. The empirical equations use the total number of jobs to determine the appropriate parameter value for any given problem structure. We do not use the total number of projects in determining the value of parameters even though it is a valuable piece of information from an industrial perspective. Note that it is the number of jobs and not the number of projects that determine the search space and drive the search algorithm.

The empirical equations developed after performing the equation may not fit the data perfectly. However, the experimentation was detailed enough to ensure that the fit obtained was indeed the best for a given problem structure. Similar experimentation was performed in each category (small, medium and large) for other parameters (Entries into Index List, Number of iterations without improvement etc) also. Based on the experimental results, the following formulae were developed:

- For fixed tabu list size, use the following formula:

    - For fixed tabu list size = .04558x – 1.0177

    - For variable tabu list size:
        - Initial size of the tabu list = 0.4426x – 0.7869
        - The Decreased size of the tabu list = -0.0254x^2 + 1.1085x – 5.9898
        - The Increased size of the tabu list = 0.0086x^3 – 0.3838x^2 + 6.0924x - 27

where x is the total number of jobs

$$\text{INT}(x) = \begin{cases} \lfloor x \rfloor, & \text{if } x \text{ is a real number with a decimal value} < 0.5 \\ \lceil x \rceil, & \text{if } x \text{ is a real number with a decimal value} \geq 0.5 \end{cases}$$

(2) **Aspiration Level (AL):** Aspiration criterion is the condition a tabu move has to satisfy in order to be released from tabu restrictions. Aspiration Level (AL) is set equal to the TWT of the initial solution at the beginning of the search process. In subsequent iterations, if the TWT of the selected best solution is less than AL then the AL is updated to be the TWT of the selected best solution. As mentioned earlier, if a tabu move results in TWT that is better than AL, the move is released from tabu restrictions and the corresponding schedule is included in the set of solutions considered for selection.

(3) **Candidate List (CL):** Candidate List consists of the best solution selected at each iteration whereas Index list entails all the local optima obtained during the search process. For a given problem instance, suppose that the initial solution gave a TWT equal to S0. The initial solution (S0) is considered to be the first local optimum and therefore it is included in the Index List as well as the Candidate List. As mentioned earlier, TWT is used as a measure of performance for any given solution. Further suppose that the best solution obtained after perturbing S0 is S1. Since S1 is best solution obtained after first iteration, it is admitted into the Candidate List. If TWT of S1 is less than TWT of S0 (S1<S0), S1 will receive a (*), which indicates that it has the potential to become a local optimum. Now let S2 be the best solution obtained after perturbing S1. S2 is admitted in the Candidate List. If TWT of S1 is less than or equal to TWT of S2 then S1 will receive another star. A solution with double stars (**) implies that it is a local optimum and is inducted into the Index List. If TWT of S2 is less than TWT of S1, then S2 will receive a star suggesting that it has the potential to become the next local optimum. Before admitting an entry into the CL, it has to be checked against all previously admitted entries to avoid duplication. If a solution already exists in the CL, the next best solution is chosen instead.

(4) **Number of iterations without improvement:** The number of iterations without improvement (IWOI) is set equal to zero before initiating the search. At subsequent iterations, if there is no improvement in the TWT value (i.e. the current TWT is equal or larger than the previous TWT), increase the IWOI by 1. The IWOI is reset to zero, if an improvement is found in TWT.

(5) **Long term memory (LTM) matrix**: The long term memory matrix is used when the tabu-search employs long term memory function for exploring the search space. The long term memory matrix is a frequency matrix that keeps track of the number of times a particular job is processed on a particular machine. The size of the LTM matrix is equal to the number of jobs times the number of machines. For example, if a problem instance has 10 jobs and 3 machines then the size of the LTM matrix would be 10 by 3 (10 rows and 3 columns = 10*3 = 30 cells). Initially, all the entries of the LTM matrix are set equal to zero. Note that not all jobs can be processed on all machines (due to machine

capability). If a job cannot be processed on a particular machine then the corresponding cells will remain empty throughout the search.

The LTM-matrix is updated after every iteration. Each cell that corresponds to the machine on which a job is processed is increased by 1, after each iteration. The LTM matrix provides information regarding the machine that is most or least frequently used by a job. We utilize this information to determine the restarting point to intensify and diversify the search process.

Step 4: There are two stopping criterion used to terminate the search: (1) Maximum number of iterations without improvement and (2) Maximum entries into the index list. Both criteria are dependent upon the size of the problem instance. Extensive experimentation was performed to determine the appropriate threshold values for IWOI Max and IL Max. IWOI Max and IL max are proportional to the total number of jobs. Hence IWOI Max and IL Max increase as the number of jobs increase.

Step 5: Tabu search employs the intensification and diversification strategies to further explore the search space. The intensification of search is carried out using the long term memory based on the maximum frequency (LTM-MAX). The diversification of search is carried out using the long term memory based on the minimum frequency (LTM-MIN). As mentioned earlier, the information pertaining to maximum and minimum frequency is stored in the LTM matrix.

The LTM search based on maximum frequency directs the search to restart from regions that have previously been considered as ‘good’. The LTM search based on the minimum frequency directs the search to restart from regions that were least or never explored. The following paragraph serves as the guideline for using the LTM matrix.

- For LTM-MAX, job-machine pair with maximum frequency from the frequency matrix is selected. Note that if one or more jobs have operation only on one machine (due to machine capability) then the cells corresponding to these jobs will have the largest entries because the job will always opt to be processed only on that machine (due to the machine capability constraint). Fixing such jobs on machines is pointless because they will be processed on the assigned machine throughout the search (regardless of whether we fix them to a particular machine or not). Extra precaution needs to be taken while selecting the job and machine pair. Only those job-machine pairs must be selected where the job (in consideration) can be processed on two or more machines. Once the job-machine pair is selected, the job is fixed to the respective machine (based on the LTM frequency matrix) throughout the search process until the next restart is invoked. If there is a tie maximum frequency, row-wise first best strategy is used to break ties.

- For LTM-MIN, job-machine pair with minimum frequency from the frequency matrix is selected. Meticulous attention is required while selecting the job-machine pair. Note that if one or more jobs have no operation on a particular machine (due to machine capability) then the cells corresponding to these jobs will have the smallest entries because the job will never opt to be processed on that machine. Therefore, we only consider those job-machine pairs in which the job (in consideration) can be processed on two or more machines. After the job-machine pair is selected, fix the job on the respective machine until the next restart is invoked. Similar to the LTM-max, use the row-wise first best strategy to break ties.

The job selected from the LTM matrix is referred to as ‘fixed job’ and the corresponding machine (on which the job is supposed to be ‘fixed’ throughout the search) is referred as ‘fixed machine’. The schedule used to restart the search is derived from the initial solution. If ‘fixed job’ is already assigned to the ‘fixed’ machine in the initial solution then the restart solution will be similar to the initial solution. In this scenario, the only difference between the initial solution and the restart solution is that the ‘fixed job’ will not be removed from the ‘fixed machine’ until another restart is invoked.

In case if the ‘fixed job’ is not already assigned to the ‘fixed machine’ in the initial solution, we insert the ‘fixed job’ to the fist position of the ‘fixed machine’. The rationale behind doing so is to address a situation when no jobs are scheduled to be processed on the ‘fixed machine’. Under this circumstance, scheduling ‘fixed job’ on ‘fixed machine’ will make the ‘fixed job’ the first as well as the only job to be processed on the ‘fixed machine’.

Using restart as a starting point, repeat Steps 1 through 4. Total number of restarts used in this research is assumed to be 2. Logendran and Sonthinen (1997), Logendran and

Subur (2000) have previously used two restarts to solve problems of industrial merit. The tabu list, Aspiration Level and iterations without improvement must be re-initialized at the beginning of each restart.

Step 6: If short term memory is utilized to commence the search, the entire search should be concluded at the end of Step 4. For long-term memory function, the search is concluded when the total number of restarts is reached (which is 2 in this case). The best solution (minimum TWT) is then chosen from the index list.

The entire algorithmic steps are written in C# using Visual Studio.NET platform. Visual Studio is an integrated development tool from Microsoft that lets developers to create user interfaces for web and windows services. Visual Studio.NET supports several different programming languages including C#.NET, VB.NET, C++, PHP and several others. The programs are written in the form of function files and are executable from C# .NET 2008. A flowchart showing the steps of the tabu search is presented in Figure 5.1.

5.6 Application of Heuristic Algorithm to Example Problem

We illustrate the application of the tabu search heuristic by means of a randomly generated sample problem, shown in Table 5.2. The problem comprises of one project and the number of jobs within the project is 11. There are a total of 5 jobs that have an operation which can be performed on more than one machine is stage 1. The due dates of jobs along with weights, release times, runtimes have been presented in Table 5.2. The sequence-dependent setup times have been included in the appendix A instead of Table 5.2 due to space limitations and overwhelming data content.

The machine availability times are also presented in Table 5.2. Note that among the 3 machines in stage 1, $M_{1,1}$ is available at t = 3, $M_{2,1}$ is available at t = 8, and $M_{3,1}$ is available at t = 4. Jobs that cannot be processed on certain machines are assumed to have a runtime of 0 (e.g. runtime of $J_{1,1}$ on $M_{3,1}$ is 0). The five methods described in Section 5.3 are applied to this example problem to obtain initial solutions.

Table 5-1 Problem Structure

Number of Jobs	11
Number of stages	17
Number of machines	19

Table 5-2 Example problem with 11 jobs

P	**J**	**Wt**	**RT**	**DD**	**$M_{1,1}$**	**$M_{2,1}$**	**$M_{3,1}$**	**$M_{1,2}$**	**$M_{1,3}$**	**$M_{1,4}$**	**$M_{1,5}$**	**$M_{1,6}$**	**$M_{1,7}$**	**$M_{1,8}$**	**$M_{1.9}$**	**$M_{1,10}$**
					3	8	4	18	22	30	36	41	48	53	61	64
								Runtime of Jobs								
1	1	1	2	1684	45	35	0	0	23	0	0	0	0	0	0	44
1	2	2	4	520	0	34	0	0	38	41	0	0	0	0	37	0
1	3	1	4	339	0	0	33	0	41	0	0	0	0	28	24	31
1	4	3	8	1462	34	0	0	32	41	0	0	0	0	0	0	0
1	5	3	4	1501	0	41	32	0	0	35	31	0	0	0	0	0
1	6	2	6	517	0	0	43	0	40	33	0	0	0	0	0	0
1	7	2	8	388	40	0	45	0	0	35	0	0	0	0	0	28
1	8	3	7	603	37	0	0	36	25	0	0	0	0	0	0	38
1	9	2	4	441	0	24	47	39	32	0	0	0	0	0	33	0
1	10	1	4	819	0	0	46	0	33	0	0	0	32	0	0	0
1	11	1	4	350	48	0	41	0	27	0	0	0	31	0	38	0

a. The following evaluations are obtained by applying the EDD method to the example problem:

- At $t = 0$, $\text{tm}_{1,1} = 3$, $tm_{2,1} = 8$, $tm_{3,1} = 4$. $NSJ = [J_{1,1}, J_{2,1}, J_{3,1}, J_{4,1}, J_{5,1}, J_{6,1}, J_{7,1}, J_{8,1}, J_{9,1}, J_{10,1}, J_{11,1}]$.
- The machine with minimum availability is $M_{1,1}$ at t = 3. $J_{1,1}$ is released at $t = 2$ hence it is selected to be processed on $M_{1,1}$. Anticipatory setup cannot be performed since the machine availability is greater than the job release time. $ST\ (J_{1,1}, M_{1,1}) = 3$, $CT\ (J_{1,1}, M_{1,1}) = 3 + 33 + 45 = 81$ (where 33 is the change over from reference to $J_{1,1}$ on $M_{1,1}$ and 45 is the runtime of $J_{1,1}$ on $M_{1,1}$). NSJ = $[J_{2,1}, J_{3,1}, J_{4,1}, J_{5,1}, J_{6,1}, J_{7,1}, J_{8,1}, J_{9,1}, J_{10,1}, J_{11,1}]$. $tm_{1,1} = 81$.
- The next machine with minimum availability is $M_{3,1}$ at $t = 4$. SJ = $[J_{3,1}, J_{5,1}, J_{9,1}, J_{10,1}, J_{11,1}]$. $J_{3,1}$ is selected since it has the minimum due date. ST $(J_{3,1}, M_{3,1}) = 4$, $CT\ (J_{3,1}, M_{3,1}) = 4 + 7 + 33 = 44$ (where 7 is the change over from reference to $J_{3,1}$ on $M_{3,1}$ and 33 is the runtime of $J_{3,1}$ on $M_{3,1}$). $NSJ = [J_{2,1}, J_{4,1}, J_{5,1}, J_{6,1}, J_{7,1}, J_{8,1}, J_{9,1}, J_{10,1}, J_{11,1}]$. $tm_{3,1} = 44$.
- The next machine with minimum availability is $M_{2,1}$ at $t = 8$. $SJ = [J_{2,1}, J_{5,1}, J_{9,1}]$. $J_{9,1}$ is selected since it has the minimum due date. $ST\ (J_{9,1}, M_{2,1}) = 8$. $CT\ (J_{9,1}, M_{2,1}) = 8 + 23 + 24 = 55$ (where 23 is the change over from reference to $J_{9,1}$ on $M_{2,1}$). $NSJ = [J_{2,1}, J_{4,1}, J_{5,1}, J_{6,1}, J_{7,1}, J_{8,1}, J_{10,1}, J_{11,1}]$. $tm_{2,1} = 55$.
- $tm_{i,k} = [81, 44, 55]$. $M_{3,1}$ has the minimum $tm_{\mathrm{i,k}}$ and $t = 44$. $SJ = [J_{5,1}, J_{6,1}, J_{7,1}, J_{10,1}, J_{11,1}]$. $J_{11,1}$ is selected since it has the minimum due date. $ST\ (J_{11,1}, M_{3,1}) = 44$, $CT\ (J_{11,1}, M_{3,1}) = 44 + 6 + 41 = 91$ (where 6 is the change over from $J_{3,1}$ to $J_{11,1}$ on $M_{3,1}$ and 41 is the runtime of $J_{11,1}$ on $M_{3,1}$). $NSJ = [J_{2,1}, J_{4,1}, J_{5,1}, J_{6,1}, J_{7,1}, J_{8,1}, J_{10,1}]$ and $tm_{3,1} = 91$.
- $tm_{i,k} = [81, 91, 55]$. $M_{2,1}$ has the minimum $tm_{i,k}$ and $t = 55$. $SJ = [J_{2,1}, J_{5,1}]$. $J_{2,1}$ is selected since it has the minimum due date. $ST\ (J_{2,1}, M_{2,1}) = 55$ and $CT\ (J_{2,1}, M_{2,1}) = 55 + 23 + 34 = 112$ (where 23 is the change over from $J_{9,1}$ to $J_{2,1}$ on $M_{2,1}$). NSJ = $[J_{4,1}, J_{5,1}, J_{6,1}, J_{7,1}, J_{8,1}, J_{10,1}]$ and $tm_{2,1} = 112$.
- $tm_{i,k} = [81, 91, 112]$. $M_{1,1}$ has the minimum $tm_{i,k}$ and $t = 81$. $SJ = [J_{4,1}, J_{7,1}, J_{8,1}]$. $J_{7,1}$ is chosen since it has the minimum due date. $ST\ (J_{7,1}, M_{1,1}) = 81$ and $CT\ (J_{7,1}, M_{1,1}) = 81 + 39 + 40 = 160$ (where 39 is the change over from $J_{1,1}$ to $J_{7,1}$

on $M_{1,1}$ and 40 is the runtime of $J_{7,1}$ on $M_{1,1}$). *NSJ* = [$J_{4,1}$, $J_{5,1}$, $J_{6,1}$, $J_{8,1}$, $J_{10,1}$] and $tm_{1,1}$ = 160.

- $tm_{i,k}$ = [160, 91, 112]. $M_{3,1}$ has the minimum $tm_{i,k}$ and t = 91. *SJ* = [$J_{5,1}$, $J_{6,1}$, $J_{10,1}$]. $J_{6,1}$ is selected since it has the minimum due date. *ST* ($J_{6,1}$, $M_{3,1}$) = 91 and *CT* ($J_{6,1}$, $M_{3,1}$) = 91+ 15 + 43 = 149 (where 15 is the change over from $J_{11,1}$ to $J_{6,1}$ on $M_{3,1}$ and 43 is the runtime of $J_{6,1}$ on $M_{3,1}$). NSJ = [$J_{4,1}$, $J_{5,1}$, $J_{8,1}$, $J_{10,1}$] and $tm_{3,1}$ = 149.
- $tm_{i,k}$ = [160, 149, 112]. $M_{2,1}$ has the minimum $tm_{\text{i,k}}$ and t = 112. *SJ* = [$J_{5,1}$]. Since $J_{5,1}$ is the only job, it is scheduled on $M_{2,1}$. *ST*($J_{5,1}$, $M_{2,1}$) = 112 and *CT*($J_{5,1}$, $M_{2,1}$) = 112 + 12 + 41 = 165. *NSJ* = [$J_{4,1}$, $J_{8,1}$, $J_{10,1}$] and $tm_{2,1}$ = 165.
- $tm_{i,k}$ = [160, 149, 165]. $M_{3,1}$ has the minimum $tm_{i,k}$ and t = 149. *SJ* = [$J_{10,1}$]. Since $J_{10,1}$ is the only job, it is scheduled on $M_{3,1}$. *ST* ($J_{10,1}$, $M_{3,1}$) = 149 and *CT* ($J_{10,1}$, $M_{3,1}$) = 149+ 3 + 46 = 198. *NSJ* = [$J_{4,1}$, $J_{8,1}$] and $tm_{3,1}$ = 198.
- $tm_{i,k}$ = [160, 198, 165]. $M_{1,1}$ has the minimum $tm_{i,k}$ and t = 160. *SJ* = [$J_{4,1}$, $J_{8,1}$]. $J_{8,1}$ is chosen since it has the minimum due date. *ST* ($J_{8,1}$, $M_{1,1}$) = 160 and *CT* ($J_{8,1}$, $M_{1,1}$) = 160 + 18 + 37 = 215 (where 18 is the change from $J_{7,1}$ to $J_{8,1}$ on $M_{1,1}$ and 37 is the runtime of $J_{8,1}$ on $M_{1,1}$). *NSJ* = [$J_{4,1}$] and $tm_{1,1}$ = 215.
- $tm_{i,k}$ = [215, 198, 165]. $M_{2,1}$ has the minimum $tm_{i,k}$ and t = 165 but since remaining unscheduled job [$J_{4,1}$] cannot be processed on $M_{2,1}$, it is excluded from future consideration.
- $tm_{i,k}$ = [215, 198]. $M_{3,1}$ has the minimum $tm_{i,k}$ and t = 198 but since remaining unscheduled job [$J_{4,1}$] cannot be processed on $M_{3,1}$, it is excluded from future consideration.
- $tm_{i,k}$ = [215]. $M_{1,1}$ has the minimum $tm_{i,k}$ and is the only machine available at t = 215. *SJ* = [$J_{4,1}$]. Since [$J_{4,1}$] is the only job, it is scheduled on $M_{1,1}$. *ST* ($J_{4,1}$, $M_{1,1}$) = 215 and *CT* ($J_{4,1}$, $M_{1,1}$) = 215 + 28 + 34 = 277 (where 28 is the change from $J_{8,1}$ to $J_{4,1}$ on $M_{1,1}$ and 34 is the runtime of $J_{4,1}$ on $M_{1,1}$). $tm_{1,1}$ = 277.
- At this time, all jobs have been processed in the first stage and are ready to be processed on the second stage. Note that not all jobs are processed in the second stage due to machine skipping. $J_{4,1}$, $J_{8,1}$ and $J_{9,1}$ are required to be processed on $M_{1,2}$. $M_{1,2}$ becomes available at t = 18 and $J_{4,1}$, $J_{8,1}$ and $J_{9,1}$ are

released at t = 277, t = 215 and t = 55, respectively. Since $J_{9,1}$ is released earliest, it is the first job to be processed on $M_{1,2}$ followed by $J_{8,1}$ and $J_{4,1}$. The start and completion time of jobs can be documented as follows.

- *ST* ($J_{9,1}$, $M_{1,2}$) = 55. *CT* ($J_{9,1}$, $M_{1,2}$) = 55 + 39 = 94 (where 39 is the run time of $J_{9,1}$ on $M_{2,1}$). Anticipatory setup is performed on the machine starting at t = 25 and finished before the release of the job (t = 55). Note that the changeover time required to change from reference to $\mathrm{J}_{9,1}$ on $\mathrm{M}_{2,1}$ is 30.
- Next, *ST* ($J_{8,1}$, $M_{1,2}$) = 215. *CT* ($J_{8,1}$, $M_{1,2}$) = 215 + 36 = 251 (where 36 is the run time of $J_{8,1}$ on $M_{2,1}$). Anticipatory setup is performed on the machine starting at t = 182 and finished before the release of the job (t = 215). Note that the changeover time required to change from $J_{9,1}$ on $M_{2,1}$ to $J_{8,1}$ on $M_{2,1}$ is 33.
- $M_{2,1}$ becomes available at t = 251. $J_{4,1}$ is the next job required to be processed on $M_{2,1}$. Setup is performed on machine starting t = 251 and completed at time 279 (changeover from $J_{8,1}$ on $M_{2,1}$ to $J_{4,1}$ on $M_{2,1}$ is 28). *ST* ($\mathrm{J}_{4,1}$, $\mathrm{M}_{1,2}$) = 279. *CT* ($\mathrm{J}_{8,1}$, $\mathrm{M}_{1,2}$) = 279 + 32 = 309 (where 32 is the run time of $J_{4,1}$ on $M_{2,1}$). At this point, there aren't any jobs that require an operation on $M_{1,2}$.
- The jobs are processed on stages 3-17 in the similar fashion. The completion time of all jobs the end of stage 17 along with their weighted tardiness is summarized in Table 5.3.

b. The following evaluations are obtained by applying (LFJ/LFM) method to the example problem:

- At t = 0, $tm_{1,1}$ = 3, $tm_{2,1}$ = 8, $tm_{3,1}$ = 4. NSJ = [$J_{1,1}$, $J_{2,1}$, $J_{3,1}$, $J_{4,1}$, $J_{5,1}$, $J_{6,1}$, $J_{7,1}$, $J_{8,1}$, $J_{9,1}$, $J_{10,1}$, $J_{11,1}$].
- The machine with minimum availability is $M_{1,1}$ at t = 3. $J_{1, 1}$ is released at t = 2 hence it is selected to be processed on $M_{1,1}$. Anticipatory setup cannot be performed since the machine availability is greater than the job release time. ST ($J_{1,1}$, $M_{1,1}$) = 3, CT ($J_{1,1}$, $M_{1,1}$) = 3 + 33+ 45 = 81 (where 33 is the change over from reference to $J_{1,1}$ on $M_{1,1}$ and 45 is the runtime of $J_{1,1}$ on $M_{1,1}$). NSJ = [$J_{2,1}$, $J_{3,1}$, $J_{4,1}$, $J_{5,1}$, $J_{6,1}$, $J_{7,1}$, $J_{8,1}$, $J_{9,1}$, $J_{10,1}$, $J_{11,1}$]. $tm_{1,1}$ = 81.
- The next machine with minimum availability is $M_{3,1}$ at t = 4. SJ = [$J_{3,1}$, $J_{5,1}$, $J_{9,1}$, $J_{10,1}$, $J_{11,1}$]. $J_{10,1}$ is selected since it is the least flexible job. ST ($J_{10,1}$, $M_{3,1}$) = 4, CT ($J_{10,1}$, $M_{3,1}$) = 4 + 9 + 46 = 59 (where 8 is the change over from reference to $J_{10,1}$ on $M_{3,1}$ and 46 is the runtime of $J_{10,1}$ on $M_{3,1}$). NSJ = [$J_{2,1}$, $J_{3,1}$, $J_{4,1}$, $J_{5,1}$, $J_{6,1}$, $J_{7,1}$, $J_{8,1}$, $J_{9,1}$, $J_{11,1}$]. $tm_{3,1}$ = 59.
- The next machine with minimum availability is $M_{2,1}$ at t = 8. SJ = [$J_{2,1}$, $J_{5,1}$, $J_{9,1}$]. $J_{2,1}$ is selected since it is the least flexible. ST ($J_{2,1}$, $M_{2,1}$) = 8. CT ($J_{2,1}$, $M_{2,1}$) = 8 + 17 + 34 = 59 (where 17 is the change over from reference to $J_{2,1}$ on $M_{2,1}$). NSJ = [$J_{3,1}$, $J_{4,1}$, $J_{5,1}$, $J_{6,1}$, $J_{7,1}$, $J_{8,1}$, $J_{9,1}$, $J_{11,1}$]. $tm_{2,1}$ = 59.
- $tm_{\mathrm{i,k}}$ = [81, 59, 59]. Though $M_{2,1}$ and $M_{3,1}$ have the same $tm_{i,k}$ (release time) at t = 59, $M_{2,1}$ is selected because it is less flexible . SJ = [$J_{5,1}$, $J_{9,1}$]. $J_{5,1}$ is selected over $J_{9,1}$ because both have same flexibility but the former has a lower job number. ST ($J_{5,1}$, $M_{3,1}$) = 59, CT ($J_{5,1}$, $M_{3,1}$) = 59 + 39 + 41 = 139 (where 39 is the change over from $J_{2,1}$ to $J_{5,1}$ on $M_{3,1}$ and 41 is the runtime of $J_{5,1}$ on $M_{3,1}$). NSJ = [$J_{3,1}$, $J_{4,1}$, $J_{6,1}$, $J_{7,1}$, $J_{8,1}$, $J_{9,1}$, $J_{11,1}$] and $tm_{3,1}$ = 139.
- $tm_{i,k}$ = [81, 139, 59]. $M_{3,1}$ has the minimum $tm_{i,k}$ and t = 59. SJ = [$J_{3,1}$, $J_{6,1}$, $J_{7,1}$, $J_{11,1}$]. $J_{6,1}$ because it is the least flexible job. ST ($J_{6,1}$, $M_{2,1}$) = 59 and CT ($J_{6,1}$, $M_{2,1}$) = 59 + 25 + 43 = 127 (where 25 is the change over from $J_{10,1}$ to $J_{6,1}$ on $M_{2,1}$). NSJ = [$J_{3,1}$, $J_{4,1}$, $J_{7,1}$, $J_{8,1}$, $J_{9,1}$, $J_{11,1}$] and $tm_{2,1}$ = 127.

- $tm_{i,k}$ = [81, 139, 127]. $M_{1,1}$ has the minimum $tm_{i,k}$ and t = 81. SJ = [$J_{4,1}$, $J_{8,1}$, $J_{11,1}$]. Though all three unscheduled jobs have same flexibility, $J_{4,1}$ is chosen because it is has the lowest job number. ST ($J_{4,1}$, $M_{1,1}$) = 81 and CT ($J_{4,1}$, $M_{1,1}$) = 81 + 31 + 34 = 146 (where 31 is the change over from $J_{1,1}$ to $J_{4,1}$ on $M_{1,1}$ and 34 is the runtime of $J_{4,1}$ on $M_{1,1}$). NSJ = [$J_{3,1}$, $J_{7,1}$, $J_{8,1}$, $J_{9,1}$, $J_{11,1}$] and $tm_{1,1}$ = 147.
- $tm_{i,k}$ = [147, 139, 127]. $M_{3,1}$ has the minimum $tm_{i,k}$ and t = 127 and SJ = [$J_{3,1}$, $J_{7,1}$, $J_{11,1}$]. $J_{3,1}$ is selected since it is least flexible. ST ($J_{3,1}$, $M_{3,1}$) = 127 and CT ($J_{3,1}$, $M_{3,1}$) = 127 + + 33 = 180 (where 20 is the change over from $J_{6,1}$ to $J_{3,1}$ on $M_{3,1}$). NSJ = [$J_{7,1}$, $J_{8,1}$, $J_{9,1}$, $J_{11,1}$] and $tm_{3,1}$ = 180.
- $tm_{i,k}$ = [147, 139, 180]. $M_{2,1}$ has the minimum $tm_{i,k}$ and t = 139. SJ = [$J_{9,1}$]. $J_{9,1}$ is scheduled to be processed on $M_{2,1}$ since it is the only remaining job . ST ($J_{9,1}$, $M_{2,1}$) = 139 and CT ($J_{8,1}$, $M_{2,1}$) = 139+ 4 + 24 = 167 where (4 is the change over from $J_{5,1}$ to $J_{9,1}$ on $M_{2,1}$ and 24 is the runtime of $J_{9,1}$ on $M_{2,1}$). NSJ = [$J_{7,1}$, $J_{8,1}$, $J_{11,1}$] and $tm_{2,1}$ = 175.
- $M_{2,1}$ is removed from further consideration since none of the unscheduled jobs can be processed on $M_{2,1}$.
- $tm_{i,k}$ = [147, 180]. $M_{1,1}$ has the minimum $tm_{i,k}$ and t = 147. SJ = [$J_{8,1}$, $J_{11,1}$]. $J_{8,1}$ is chosen since it is less flexible than $J_{11,1}$. ST($J_{8,1}$, $M_{1,1}$) = 147 and CT($J_{8,1}$, $M_{1,1}$) = 147+ 9 + 37 = 193 (where 9 is the change over from $\mathrm{J}_{4,1}$ to $\mathrm{J}_{8,1}$ on $M_{1,1}$ and 37 is the runtime of $J_{11,1}$ on $M_{1,1}$). NSJ = [$J_{7,1}$, $J_{11,1}$] and $tm_{1,1}$ = 193.
- $tm_{i,k}$ = [193, 180]. $M_{3,1}$ has the minimum $tm_{i,k}$ and t = 180. SJ = [$J_{7,1,}$ $J_{10,1}$]. $J_{7,1}$ is selected since it is the only job that can be on $M_{3,1}$. ST ($J_{7,1}$, $M_{3,1}$) = 180 and CT ($J_{7,1}$, $M_{3,1}$) = 180 + 12 + 45 = 237 (where 12 is the change over from $J_{3,1}$ to $J_{7,1}$ on $M_{3,1}$ and 45 is the runtime of $J_{7,1}$ on $M_{3,1}$). NSJ = [$J_{10,1}$] and $tm_{3,1}$ = 237.
- $tm_{\mathrm{i,k}}$ = [193, 237]. $M_{1,1}$ has the minimum $tm_{\mathrm{i,k}}$ and t = 193. SJ = [$J_{11,1}$]. $J_{11,1}$ is the only remaining unscheduled job. ST ($J_{11,1}$, $M_{1,1}$) = 193 and CT ($J_{4,1}$, $M_{1,1}$) = 193 + 8 + 48 = 249 (where 8 is the change from $J_{8,1}$ to $J_{11,1}$ on $M_{1,1}$ and 48 is the runtime of $J_{11,1}$ on $M_{1,1}$). NSJ = [Ø]

- At this time, all jobs have been processed in the first stage and are ready to be processed on the second stage. Note that not all jobs are processed in the following stages (stage 2-17) due to machine skipping.
- The jobs are processed on stages 2-17 in the similar fashion. The completion time of all jobs the end of stage 17 along with their weighted tardiness is summarized in Table 5.3.

c. The following evaluations are obtained by applying IS3 method to the example problem:

- At t = 0, $tm_{1,1}$ = 3, $tm_{2,1}$ = 8, $tm_{3,1}$ = 4. NSJ = [$J_{1,1}$, $J_{2,1}$, $J_{3,1}$, $J_{4,1}$, $J_{5,1}$, $J_{6,1}$, $J_{7,1}$, $J_{8,1}$, $J_{9,1}$, $J_{10,1}$, $J_{11,1}$].
- The machine with minimum availability is $M_{1,1}$ at t = 3. $J_{1,\,1}$ is released at t = 2 hence it is selected to be processed on $M_{1,1}$. Anticipatory setup cannot be performed since the machine availability is greater than the job release time. ST ($J_{1,1}$, $M_{1,1}$) = 3, CT ($J_{1,1}$, $M_{1,1}$) = 3 + 33+ 45 = 81 (where 33 is the change over from reference to $J_{1,1}$ on $M_{1,1}$ and 45 is the runtime of $J_{1,1}$ on $M_{1,1}$). NSJ = [$J_{2,1}$, $J_{3,1}$, $J_{4,1}$, $J_{5,1}$, $J_{6,1}$, $J_{7,1}$, $J_{8,1}$, $J_{9,1}$, $J_{10,1}$, $J_{11,1}$]. $tm_{1,1}$ = 81.
- The next machine with minimum availability is $M_{3,1}$ at t = 4. SJ = [$J_{3,1}$, $J_{5,1}$, $J_{9,1}$, $J_{10,1}$, $J_{11,1}$]. $J_{5,1}$ is selected since it has the minimum tardiness. ST ($J_{5,1}$, $M_{3,1}$) = 4, CT ($J_{5,1}$, $M_{3,1}$) = 4 + 13 + 41 = 58 (where 13 is the change over from reference to $J_{5,1}$ on $M_{3,1}$ and 41 is the runtime of $J_{5,1}$ on $M_{3,1}$). NSJ = [$J_{2,1}$, $J_{3,1}$, $J_{4,1}$, $J_{6,1}$, $J_{7,1}$, $J_{8,1}$, $J_{9,1}$, $J_{10,1}$, $J_{11,1}$]. $tm_{3,1}$ = 58.
- The next machine with minimum availability is $M_{2,1}$ at t = 8. SJ = [$J_{2,1}$, $J_{5,1}$, $J_{9,1}$]. $J_{2,1}$ is selected since it has the minimum tardiness. ST ($J_{2,1}$, $M_{2,1}$) = 8. CT ($J_{2,1}$, $M_{2,1}$) = 8 + 17 + 34 = 59 (where 17 is the change over from reference to $J_{2,1}$ on $M_{2,1}$). NSJ = [$J_{3,1}$, $J_{4,1}$, $J_{6,1}$, $J_{7,1}$, $J_{8,1}$, $J_{9,1}$, $J_{10,1}$, $J_{11,1}$]. $tm_{2,1}$ = 59.
- $tm_{i,k}$ = [81, 58, 59]. $M_{3,1}$ has the minimum $tm_{i,k}$ and t = 58. SJ = [$J_{3,1}$, $J_{6,1}$, $J_{7,1}$, $J_{9,1}$, $J_{10,1}$, $J_{11,1}$]. $J_{10,1}$ is selected since it has the minimum tardiness. ST ($J_{10,1}$, $M_{3,1}$) = 58, CT ($J_{10,1}$, $M_{3,1}$) = 58 + 4 + 46 = 108 (where 4 is the change over from $J_{5,1}$ to $J_{10,1}$ on $M_{3,1}$ and 46 is the runtime of $J_{10,1}$ on $M_{3,1}$). NSJ = [$J_{3,1}$, $J_{4,1}$, $J_{6,1}$, $J_{7,1}$, $J_{8,1}$, $J_{9,1}$, $J_{11,1}$] and $tm_{3,1}$ = 108.
- $tm_{i,k}$ = [81, 108, 59]. $M_{2,1}$ has the minimum $tm_{i,k}$ and t = 59. SJ = [$J_{9,1}$]. $J_{9,1}$ is selected since it is the only remaining job that can be processed on $M_{2,1}$. ST ($J_{9,1}$, $M_{2,1}$) = 59 and CT ($J_{9,1}$, $M_{2,1}$) = 59 + 12 + 24 = 95 (where 12 is the change over from $J_{2,1}$ to $J_{9,1}$ on $M_{2,1}$). NSJ = [$J_{3,1}$, $J_{4,1}$, $J_{6,1}$, $J_{7,1}$, $J_{8,1}$, $J_{11,1}$]and $tm_{2,1}$ = 95.

- $tm_{i,k}$ = [81, 108, 95]. $M_{1,1}$ has the minimum $tm_{i,k}$ and t = 81. SJ = [$J_{4,1}$, $J_{7,1}$, $J_{8,1}$, $J_{11,1}$]. $J_{4,1}$ is chosen since it has the minimum tardiness. ST ($J_{4,1}$, $M_{1,1}$) = 81 and CT ($J_{4,1}$, $M_{1,1}$) = 81 + 31 + 34 = 146 (where 31 is the change over from $J_{1,1}$ to $J_{4,1}$ on $M_{1,1}$ and 34 is the runtime of $J_{4,1}$ on $M_{1,1}$). NSJ = [$J_{3,1}$, $J_{6,1}$, $J_{7,1}$, $J_{8,1}$, $J_{11,1}$] and $tm_{1,1}$ = 146.
- $tm_{i,k}$ = [146, 108, 95]. $M_{2,1}$ has the minimum $tm_{i,k}$ and t = 95. SJ = [Ø]. Thus $M_{2,1}$ is not taken into consideration and is removed from the set of available machines.
- $tm_{i,k}$ = [146, 108]. $M_{3,1}$ has the minimum $tm_{i,k}$ and t = 108. SJ = [$J_{3,1}$, $J_{6,1}$, $J_{7,1}$, $J_{11,1}$]. $J_{6,1}$ is selected since it has the minimum tardiness. ST($J_{6,1}$, $M_{3,1}$) = 108 and CT($J_{6,1}$, $M_{3,1}$) = 108 + 33 + 35 = 176. (where 33 is the changeover from $J_{10,1}$ to $J_{6,1}$ on $M_{3,1}$). NSJ = [$J_{3,1}$, $J_{7,1}$, $J_{8,1}$, $J_{11,1}$] and $tm_{,1}$ = 176
- $tm_{i,k}$ = [146, 176]. $M_{1,1}$ has the minimum $tm_{i,k}$ and t = 146. SJ = [$J_{8,1}$, $J_{11,1}$]. $J_{8,1}$ is chosen since it has the minimum tardiness. ST ($J_{8,1}$, $M_{1,1}$) = 146 and CT ($J_{8,1}$, $M_{1,1}$) = 146+ 10 + 37 = 193 (where 10 is the changeover from $J_{4,1}$ to $J_{8,1}$ on $M_{1,1}$). NSJ = [$J_{3,1}$, $J_{7,1}$, $J_{11,1}$] and $tm_{1,1}$ = 193.
- $tm_{i,k}$ = [193, 176]. $M_{3,1}$ has the minimum $tm_{i,k}$ and t = 176. SJ = [$J_{3,1}$, $J_{7,1}$, $J_{11,1}$]. $J_{7,1}$ is chosen since it has the minimum tardiness. ST ($J_{7,1}$, $M_{3,1}$) = 176 and CT ($J_{7,1}$, $M_{3,1}$) = 176 + 14 + 45 = 235 (where 14 is the change from $J_{6,1}$ to $J_{7,1}$ on $M_{3,1}$ and 45 is the runtime of $J_{7,1}$ on $M_{3,1}$). NSJ = [$J_{3,1}$, $J_{11,1}$] and $tm_{3,1}$ = 235.
- $tm_{i,k}$ = [193, 235]. $M_{1,1}$ has the minimum $tm_{i,k}$ and t = 193. SJ = [$J_{11,1}$]. Since $J_{11,1}$ is the only remaining job that can be processed on $M_{1,1}$, it is scheduled on $M_{1,1}$. ST ($J_{11,1}$, $M_{1,1}$) = 193 and CT ($J_{11,1}$, $M_{1,1}$) = 193 + 8 + 48 = 249 (where 8 is the change from $J_{8,1}$ to $J_{11,1}$ on $M_{1,1}$ and 48 is the runtime of $J_{11,1}$ on $M_{1,1}$). $tm_{1,1}$ = 249. NSJ = [$J_{3,1}$].
- $tm_{i,k}$ = [249, 235]. $M_{3,1}$ has the minimum $tm_{i,k}$ and t = 235. SJ = [$J_{3,1}$]. The only job that can be processed on $M_{3,1}$ is $J_{3,1}$. Therefore, it is scheduled to be processed on $M_{3,1}$. ST ($J_{3,1}$, $M_{3,1}$) = 235 and CT ($J_{3,1}$, $M_{3,1}$) = 235 + 6 + 33 = 274 (where 6 is the change from $J_{6,1}$ to $J_{3,1}$ on $M_{3,1}$ and 33 is the runtime of $J_{3,1}$ on $M_{3,1}$). $tm_{1,1}$ = 274 and NSJ = [Ø].

- At this time, all jobs have been processed in the first stage and are ready to be processed on the second stage. Note that not all jobs are processed in the second stage due to machine skipping. $J_{4,1}$, $J_{8,1}$ and $J_{9,1}$ are required to be processed on $M_{1,2}$. $M_{1,2}$ becomes available at t = 18 and $J_{4,1}$, $J_{8,1}$ and $J_{9,1}$ are released at t = 146, t = 193 and t = 95 respectively. Since $J_{9,1}$ is released earliest, it is the first job to be processed on $M_{1,2}$ followed by $J_{4,1}$ and $J_{8,1}$. The start and completion time of jobs can be documented as follows.
- ST ($J_{9,1}$, $M_{1,2}$) = 55. *CT* ($J_{9,1}$, $M_{1,2}$) = 95 + 39 = 134 (where 39 is the run time of $J_{9,1}$ on $M_{2,1}$) Anticipatory setup is performed on the machine starting at t = 65 and finished before the release of the job (t = 95). Note that the changeover time required to change from reference to $J_{9,1}$ on $M_{2,1}$ is 30.
- $M_{2,1}$ becomes available at t = 134. $J_{4,1}$ is the next job required to be processed on $M_{2,1}$. Anticipatory setup is performed on machine starting at t = 134 and completed at time 159 (changeover from $J_{9,1}$ on $M_{2,1}$ to $J_{4,1}$ on $M_{2,1}$ is 25). *ST* ($J_{4,1}$, $M_{1,2}$) = 159. *CT* ($J_{4,1}$, $M_{1,2}$) = 159 + 32 = 191 (where 32 is the run time of $J_{4,1}$ on $M_{2,1}$).
- Next, $M_{2,1}$ becomes available at t = 191. Anticipatory setup is performed on the machine starting at t = 191 and finished at t = 224. Note that the changeover time required to change from $J_{4,1}$ on $M_{2,1}$ to $J_{8,1}$ on $M_{2,1}$ is 33). *ST*($J_{8,1}$, $M_{1,2}$) = 234. *CT*($J_{8,1}$, $M_{1,2}$) = 234 + 36 = 260 (where 36 is the run time of $J_{8,1}$ on $M_{2,1}$).
- The jobs are processed on stages 3-17 in the similar fashion. The completion time of all jobs the end of stage 17 along with their weighted tardiness is summarized in Table 5.3.

d. The following evaluations are obtained by applying (Due Date/Weight) method to the example problem:

- At t = 0, $tm_{1,1}$ = 3, $tm_{2,1}$ = 8, $tm_{3,1}$ = 4. *NSJ* = [$J_{1,1}$, $J_{2,1}$, $J_{3,1}$, $J_{4,1}$, $J_{5,1}$, $J_{6,1}$, $J_{7,1}$, $J_{8,1}$, $J_{9,1}$, $J_{10,1}$, $J_{11,1}$].
- The machine with minimum availability is $M_{1,1}$ at t = 3. $J_{1,\,1}$ is released at t = 2 hence it is selected to be processed on $M_{1,1}$. Anticipatory setup cannot be performed since the machine availability is greater than the job release time. *ST* ($J_{1,1}$, $M_{1,1}$) = 3, *CT* ($J_{1,1}$, $M_{1,1}$) = 3 + 33+ 45 = 81 (where 33 is the change over from reference to $J_{1,1}$ on $M_{1,1}$ and 45 is the runtime of $J_{1,1}$ on $M_{1,1}$). *NSJ* = [$J_{2,1}$, $J_{3,1}$, $J_{4,1}$, $J_{5,1}$, $J_{6,1}$, $J_{7,1}$, $J_{8,1}$, $J_{9,1}$, $J_{10,1}$, $J_{11,1}$]. $tm_{1,1}$ = 81.
- The next machine with minimum availability is $M_{3,1}$ at t = 4. *SJ* = [$J_{3,1}$, $J_{5,1}$, $J_{9,1}$, $J_{10,1}$, $J_{11,1}$]. $J_{9,1}$ is selected since it has the minimum (due date/weight) ratio. *ST* ($J_{9,1}$, $M_{3,1}$) = 4, *CT* ($J_{9,1}$, $M_{3,1}$) = 4 + 2 + 47 = 53 (where 2 is the change over from reference to $J_{9,1}$ on $M_{3,1}$ and 47 is the runtime of $J_{9,1}$ on $M_{3,1}$). *NSJ* = [$J_{2,1}$, $J_{3,1}$, $J_{4,1}$, $J_{5,1}$, $J_{6,1}$, $J_{7,1}$, $J_{8,1}$, $J_{10,1}$, $J_{11,1}$]. $tm_{3,1}$ = 53.
- The next machine with minimum availability is $M_{2,1}$ at t = 8. *SJ* = [$J_{2,1}$, $J_{5,1}$]. $J_{2,1}$ is selected since it has the minimum (due date/weight). *ST* ($J_{2,1}$, $M_{2,1}$) = 8. *CT* ($J_{2,1}$, $M_{2,1}$) = 8 + 17 + 34 = 59 (where 17 is the change over from reference to $J_{2,1}$ on $M_{2,1}$). *NSJ* = [$J_{3,1}$, $J_{4,1}$, $J_{5,1}$, $J_{6,1}$, $J_{7,1}$, $J_{8,1}$, $J_{10,1}$, $J_{11,1}$]. $tm_{2,1}$ = 59.
- $tm_{\mathrm{i,k}}$ = [81, 53, 59]. $M_{3,1}$ has the minimum $tm_{i,k}$ and t = 53. *SJ* = [$J_{3,1}$, $J_{5,1}$, $J_{6,1}$, $J_{7,1}$, $J_{10,1}$, $J_{11,1}$]. $J_{7,1}$ is selected since it has the minimum due date/weight ratio. *ST* ($J_{7,1}$, $M_{3,1}$) = 53, *CT* ($J_{7,1}$, $M_{3,1}$) = 53 + 16 + 45 = 114 (where 16 is the change over from $J_{9,1}$ to $J_{7,1}$ on $M_{3,1}$ and 45 is the runtime of $J_{7,1}$ on $M_{3,1}$). *NSJ* = [$J_{3,1}$, $J_{4,1}$, $J_{5,1}$, $J_{6,1}$, $J_{8,1}$, $J_{10,1}$, $J_{11,1}$]and $tm_{3,1}$ = 114.
- $tm_{i,k}$ = [81, 114, 59]. $M_{2,1}$ has the minimum $tm_{i,k}$ and t = 59. *SJ* = [$J_{5,1}$]. $J_{5,1}$ is selected since it is the only job that can be processed on $M_{2,1}$. *ST* ($J_{5,1}$, $M_{2,1}$) = 59 and *CT* ($J_{5,1}$, $M_{2,1}$) = 59 + 12 + 41 = 112 (where 12 is the change over from $J_{2,1}$ to $J_{5,1}$ on $M_{2,1}$). *NSJ* = [$J_{3,1}$, $J_{4,1}$, $J_{6,1}$, $J_{8,1}$, $J_{10,1}$, $J_{11,1}$] and $tm_{2,1}$ = 112.
- $tm_{i,k}$ = [81, 114, 112]. $M_{1,1}$ has the minimum $tm_{i,k}$ and t = 81. *SJ* = [$J_{4,1}$, $J_{8,1}$, $J_{11,1}$]. $J_{8,1}$ is chosen since it has the minimum due date/weight. *ST* ($J_{8,1}$, $M_{1,1}$) =

81 and *CT* ($J_{8,1}$, $M_{1,1}$) = 81 + 29 + 37 = 147 (where 29 is the change over from $J_{1,1}$ to $J_{8,1}$ on $M_{1,1}$ and 37 is the runtime of $J_{8,1}$ on $M_{1,1}$). *NSJ* = [$J_{3,1}$, $J_{4,1}$, $J_{6,1}$,$J_{10,1}$, $J_{11,1}$] and $tm_{1,1}$ = 147.

- $tm_{i,k}$ = [147, 114, 112]. $M_{2,1}$ has the minimum $tm_{i,k}$ and *t* = 112 but since the remaining unscheduled jobs cannot be processed on $M_{2,1}$, they are excluded from future consideration.
- $tm_{\mathrm{i,k}}$ = [147, 114]. $M_{3,1}$ has the minimum $tm_{i,k}$ and *t* = 114. *SJ* = [$J_{3,1}$, $J_{6,1}$, $J_{10,1}$, $J_{11,1}$]. $J_{6,1}$ is selected since it has the minimum due date/weight ratio . *ST* ($J_{6,1}$, $M_{3,1}$) = 114 and *CT* ($J_{6,1}$, $M_{3,1}$) = 114+ 18 + 43 = 175 where (18 is the change over from $J_{7,1}$ to $J_{6,1}$ on $M_{3,1}$ and 43 is the runtime of $J_{6,1}$ on $M_{3,1}$). NSJ = [$J_{3,1}$, $J_{4,1}$, $J_{10,1}$, $J_{11,1}$]and $tm_{3,1}$ = 175.
- $tm_{i,k}$ = [147, 175]. $M_{1,1}$ has the minimum $tm_{i,k}$ and t = 147. *SJ* = [$J_{11,1}$, $J_{4,1}$]. $J_{11,1}$ is chosen since it has the minimum Due Date/weight ratio. *ST*($J_{11,1}$, $M_{1,1}$) = 147 and *CT*($J_{11,1}$, $M_{1,1}$) = 147 + 8 + 48 = 203 (where 8 is the change over from $J_{8,1}$ to $J_{11,1}$ on $M_{1,1}$ and 48 is the runtime of $J_{11,1}$ on $\mathrm{M}_{1,1}$). *NSJ* = [$J_{3,1}$, $J_{4,1}$, $J_{10,1}$] and $tm_{1,1}$ = 203.
- $tm_{i,k}$ = [203, 175]. $M_{3,1}$ has the minimum $tm_{i,k}$ and *t* = 175. *SJ* = [$J_{3,1}$,$J_{10,1}$]. Since $J_{3,1}$ is selected since it has the minimum Due Date/weight ratio and is scheduled on $M_{3,1}$. *ST* ($J_{3,1}$, $M_{3,1}$) = 175 and *CT* ($J_{3,1}$, $M_{3,1}$) = 175 + 20 + 33 = 228 (where 20 is the change over from $J_{6,1}$ to $J_{3,1}$ on $M_{3,1}$ and 33 is the runtime of $J_{3,1}$ on $M_{3,1}$). *NSJ* = [$J_{4,1}$, $J_{10,1}$] and $tm_{3,1}$ = 228.
- $tm_{i,k}$ = [203, 228]. $M_{1,1}$ has the minimum $tm_{\mathrm{i,k}}$ and *t* = 203. *SJ* = [$J_{4,1}$]. $J_{4,1}$ is only remaining unscheduled job. *ST* ($J_{4,1}$, $M_{1,1}$) = 203 and *CT* ($J_{4,1}$, $M_{1,1}$) = 203 + 4 + 34 = 241 (where 4 is the change from $J_{11,1}$ to $J_{4,1}$ on $M_{1,1}$ and 34 is the runtime of $J_{4,1}$ on $M_{1,1}$). *NSJ* = [$J_{10,1}$] and $tm_{1,1}$ = 241.
- $tm_{i,k}$ = [241, 228]. $M_{3,1}$ has the minimum $tm_{i,k}$ and *t* = 228. *SJ* = [$J_{10,1}$]. Since $J_{10,1}$ is the only remaining unscheduled job, [$J_{10,1}$] is scheduled to be processed on $M_{3,1}$. *ST* ($J_{10,1}$, $M_{3,1}$) = 228 and *CT* ($J_{10,1}$, $M_{3,1}$) = 228 + 34 + 46 = 308. *NSJ* = [Ø] and $tm_{3,1}$ = 308.
- At this time, all jobs have been processed in the first stage and are ready to be processed on the second stage. Note that not all jobs are processed in the

second stage due to machine skipping. $J_{4,1}$, $J_{8,1}$ and $J_{9,1}$ are required to be processed on $M_{1,2}$. $M_{1,2}$ becomes available at t = 18 and $J_{4,1}$, $J_{8,1}$ and $J_{9,1}$ are released at t = 241, t = 147 and t = 53 respectively. Since $J_{9,1}$ is released earliest, it is the first job to be processed on $M_{1,2}$ followed by $J_{8,1}$ and $J_{4,1}$. The start and completion time of jobs can be documented as follows.

- *ST* ($J_{9,1}$, $M_{1,2}$) = 53. *CT* ($J_{9,1}$, $M_{1,2}$) = 53 + 39 = 92 (where 39 is the run time of $J_{9,1}$ on $M_{2,1}$). Anticipatory setup is performed on the machine starting at t = 23 and finished before the release of the job (t = 53). Note that the changeover time required to change from reference to $J_{9,1}$ on $M_{2,1}$ is 30..
- Next, *ST* ($J_{8,1}$, $M_{1,2}$) = 147. *CT* ($J_{8,1}$, $M_{1,2}$) = 147 + 36 = 183 (where 36 is the run time of $J_{8,1}$ on $M_{2,1}$). Anticipatory setup is performed on the machine starting at t = 114 and finished before the release of the job (t = 147). Note that the changeover time required to change from $J_{9,1}$ on $M_{2,1}$ to $J_{8,1}$ on $M_{2,1}$ is 33.
- $M_{2,1}$ becomes available at t = 183. $J_{4,1}$ is the next job required to be processed on $M_{2,1}$. Anticipatory setup is performed on machine starting at t = 251 and completed at time 213 (changeover from $J_{8,1}$ on $M_{2,1}$ to $J_{4,1}$ on $M_{2,1}$ is 28). *ST* ($J_{4,1}$, $M_{1,2}$) = 241. *CT* ($J_{8,1}$, $M_{1,2}$) = 241 + 32 = 273 (where 32 is the run time of $J_{4,1}$ on $M_{2,1}$). At this point, there aren't any jobs that require an operation on $M_{1,2}$.
- The jobs are processed on stages 3-17 in the similar fashion. The completion time of all jobs the end of stage 17 along with their weighted tardiness is summarized in Table 5.3.

e. The following evaluations are obtained by applying (HCR-Hybrid Critical Ratio) method to the example problem:

- At $t = 0$, $tm_{1,1} = 3$, $tm_{2,1} = 8$, $tm_{3,1} = 4$. $NSJ = [J_{1,1}, J_{2,1}, J_{3,1}, J_{4,1}, J_{5,1}, J_{6,1}, J_{7,1}, J_{8,1}, J_{9,1}, J_{10,1}, J_{11,1}]$.
- The machine with minimum availability is $M_{1,1}$ at $t = 3$. $J_{1, 1}$ is released at $t = 2$ hence it is selected to be processed on $M_{1,1}$. Anticipatory setup cannot be performed since the machine availability is greater than the job release time. $ST\ (J_{1,1}, M_{1,1}) = 3$, $CT\ (J_{1,1}, M_{1,1}) = 3 + 33 + 45 = 81$ (where 33 is the change over from reference to $J_{1,1}$ on $M_{1,1}$ and 45 is the runtime of $J_{1,1}$ on $M_{1,1}$). $NSJ = [J_{2,1}, J_{3,1}, J_{4,1}, J_{5,1}, J_{6,1}, J_{7,1}, J_{8,1}, J_{9,1}, J_{10,1}, J_{11,1}]$. $tm_{1,1} = 81$.
- The next machine with minimum availability is $M_{3,1}$ at $t = 4$. $SJ = [J_{3,1}, J_{5,1}, J_{9,1}, J_{10,1}, J_{11,1}]$. $J_{9,1}$ is selected since it has the minimum HCR ratio. $ST\ (J_{9,1}, M_{3,1}) = 4$, $CT\ (J_{9,1}, M_{3,1}) = 4 + 2 + 47 = 53$ (where 2 is the change over from reference to $J_{9,1}$ on $M_{3,1}$ and 47 is the runtime of $J_{9,1}$ on $M_{3,1}$). $NSJ = [J_{2,1}, J_{3,1}, J_{4,1}, J_{5,1}, J_{6,1}, J_{7,1}, J_{8,1}, J_{10,1}, J_{11,1}]$. $tm_{3,1} = 53$.
- The next machine with minimum availability is $M_{2,1}$ at $t = 8$. $SJ = [J_{2,1}, J_{5,1}]$. $J_{2,1}$ is selected since it has the minimum HCR ratio. $ST\ (J_{2,1}, M_{2,1}) = 8$. $CT\ (J_{2,1}, M_{2,1}) = 8 + 17 + 34 = 59$ (where 17 is the change over from reference to $J_{2,1}$ on $M_{2,1}$). $NSJ = [J_{3,1}, J_{4,1}, J_{5,1}, J_{6,1}, J_{7,1}, J_{8,1}, J_{10,1}, J_{11,1}]$. $tm_{2,1} = 59$.
- $tm_{\mathrm{i,k}} = [81, 53, 59]$. $M_{3,1}$ has the minimum $tm_{i,k}$ and $t = 53$. $SJ = [J_{3,1}, J_{5,1}, J_{6,1}, J_{7,1}, J_{10,1}, J_{11,1}]$. $J_{7,1}$ is selected since it has the minimum HCR ratio. $ST\ (J_{7,1}, M_{3,1}) = 53$, $CT\ (J_{7,1}, M_{3,1}) = 53 + 16 + 45 = 114$ (where 16 is the change over from $J_{9,1}$ to $J_{7,1}$ on $M_{3,1}$ and 45 is the runtime of $J_{7,1}$ on $M_{3,1}$). $NSJ = [J_{3,1}, J_{4,1}, J_{5,1}, J_{6,1}, J_{8,1}, J_{10,1}, J_{11,1}]$ and $tm_{3,1} = 114$.
- $tm_{i,k} = [81, 114, 59]$. $M_{2,1}$ has the minimum $tm_{i,k}$ and $t = 59$. $SJ = [J_{5,1}]$. $J_{5,1}$ is selected since it is the only job that can be processed on $M_{2,1}$. $ST\ (J_{5,1}, M_{2,1}) = 59$ and $CT\ (J_{5,1}, M_{2,1}) = 59 + 12 + 41 = 112$ (where 12 is the change over from $J_{2,1}$ to $J_{5,1}$ on $M_{2,1}$). $NSJ = [J_{3,1}, J_{4,1}, J_{6,1}, J_{8,1}, J_{10,1}, J_{11,1}]$ and $tm_{2,1} = 112$.
- $tm_{i,k} = [81, 114, 112]$. $M_{1,1}$ has the minimum $tm_{i,k}$ and $t = 81$. $SJ = [J_{4,1}, J_{8,1}, J_{11,1}]$. $J_{8,1}$ is chosen since it has the minimum HCR ratio. $ST\ (J_{8,1}, M_{1,1}) = 81$

and CT ($J_{8,1}$, $M_{1,1}$) = 81 + 29 + 37 = 147 (where 29 is the change over from $J_{1,1}$ to $J_{8,1}$ on $M_{1,1}$ and 37 is the runtime of $J_{8,1}$ on $M_{1,1}$). NSJ = [$J_{3,1}$, $J_{4,1}$, $J_{6,1}$,$J_{10,1}$, $J_{11,1}$] and $tm_{1,1}$ = 147.

- $tm_{i,k}$ = [147, 114, 112]. $M_{2,1}$ has the minimum $tm_{i,k}$ and t = 112 but since remaining unscheduled jobs cannot be processed on $M_{2,1}$, it is excluded from future consideration.
- $tm_{i,k}$ = [147, 114]. $M_{3,1}$ has the minimum $tm_{i,k}$ and t = 114. SJ = [$J_{3,1}$, $J_{6,1}$, $J_{10,1}$, $J_{11,1}$]. $J_{6,1}$ is selected since it has the minimum HCR ratio . ST ($J_{6,1}$, $M_{3,1}$) = 114 and CT ($J_{6,1}$, $M_{3,1}$) = 114+ 18 + 43 = 175 where (18 is the change over from $J_{7,1}$ to $J_{6,1}$ on $M_{3,1}$ and 43 is the runtime of $J_{6,1}$ on $M_{3,1}$). NSJ = [$J_{3,1}$, $J_{4,1}$, $J_{10,1}$, $J_{11,1}$]and $tm_{3,1}$ = 175.
- $tm_{i,k}$ = [147, 175]. $M_{1,1}$ has the minimum $tm_{i,k}$ and t = 147. SJ = [$J_{11,1}$, $J_{4,1}$]. $J_{11,1}$ is chosen since it has the minimum HCR ratio. ST($J_{11,1}$, $M_{1,1}$) = 147 and CT($J_{11,1}$, $M_{1,1}$) = 147 + 8 + 48 = 203 (where 8 is the change over from $\mathrm{J}_{8,1}$ to $\mathrm{J}_{11,1}$ on $M_{1,1}$ and 48 is the runtime of $J_{11,1}$ on $M_{1,1}$). NSJ = [$J_{3,1}$, $J_{4,1}$, $J_{10,1}$] and $tm_{1,1}$ = 203.
- $tm_{i,k}$ = [203, 175]. $M_{3,1}$ has the minimum $tm_{i,k}$ and t = 175. SJ = [$J_{3,1}$,$J_{10,1}$]. Since $J_{3,1}$ is selected since it has the minimum HCR ratio and is scheduled on $M_{3,1}$. ST ($J_{3,1}$, $M_{3,1}$) = 175 and CT ($J_{3,1}$, $M_{3,1}$) = 175 + 20 + 33 = 228 (where 20 is the change over from $J_{6,1}$ to $J_{3,1}$ on $M_{3,1}$ and 33 is the runtime of $J_{3,1}$ on $M_{3,1}$). NSJ = [$J_{4,1}$, $J_{10,1}$] and $tm_{3,1}$ = 228.
- $tm_{\mathrm{i,k}}$ = [203, 228]. $M_{1,1}$ has the minimum $tm_{\mathrm{i,k}}$ and t = 203. SJ = [$J_{4,1}$]. $J_{4,1}$ is the only remaining unscheduled job. ST ($J_{4,1}$, $M_{1,1}$) = 203 and CT ($J_{4,1}$, $M_{1,1}$) = 203 + 4 + 34 = 241 (where 4 is the change from $J_{11,1}$ to $J_{4,1}$ on $M_{1,1}$ and 34 is the runtime of $J_{4,1}$ on $M_{1,1}$). NSJ = [$J_{10,1}$] and $tm_{1,1}$ = 241.
- $tm_{\mathrm{i,k}}$ = [241, 228]. $M_{3,1}$ has the minimum $tm_{\mathrm{i,k}}$ and t = 228. SJ = [$J_{10,1}$]. Since $J_{10,1}$ is the only remaining unscheduled job, [$J_{10,1}$] is scheduled to be processed on processed on $M_{3,1}$. ST ($J_{10,1}$, $M_{3,1}$) = 228 and CT ($J_{10,1}$, $M_{3,1}$) = 228 + 34 + 46 = 308. NSJ = [Ø] and $tm_{3,1}$ = 308.
- At this time, all jobs have been processed in the first stage and are ready to be processed on the second stage. Note that not all jobs are processed in the

second stage due to machine skipping. $J_{4,1}$, $J_{8,1}$ and $J_{9,1}$ are required to be processed on $M_{1,2}$. $M_{1,2}$ becomes available at t = 18 and $J_{4,1}$, $J_{8,1}$ and $J_{9,1}$ are released at t = 241, t = 147 and t = 53 respectively. Since $J_{9,1}$ is released earliest, it is the first job to be processed on $M_{1,2}$ followed by $J_{8,1}$ and $J_{4,1}$. The start and completion times of jobs can be documented as follows.

- *ST* ($J_{9,1}$, $M_{1,2}$) = 53. *CT* ($J_{9,1}$, $M_{1,2}$) = 53 + 39 = 92 (where 39 is the run time of $J_{9,1}$ on $M_{2,1}$) Anticipatory setup is performed on the machine starting t = 23 and finished before the release of the job (t = 53). Note that the changeover time required to change from reference to $J_{9,1}$ on $M_{2,1}$ is 30.
- Next, *ST* ($J_{8,1}$, $M_{1,2}$) = 147. *CT* ($J_{8,1}$, $M_{1,2}$) = 147 + 36 = 183 (where 36 is the run time of $J_{8,1}$ on $M_{2,1}$). Anticipatory setup is performed on the machine starting t = 114 and finished before the release of the job (t = 147). Note that the changeover time required to change from $J_{9,1}$ on $M_{2,1}$ to $J_{8,1}$ on $M_{2,1}$ is 33..
- $M_{2,1}$ becomes available at t = 183. $J_{4,1}$ is the next job required to be processed on $M_{2,1}$. Anticipatory setup is performed on machine starting at t = 251 and completed at time 213 (changeover from $J_{8,1}$ on $M_{2,1}$ to $J_{4,1}$ on $M_{2,1}$ is 28). *ST* ($J_{4,1}$, $M_{1,2}$) = 241. *CT* ($J_{8,1}$, $M_{1,2}$) = 241 + 32 = 273 (where 32 is the run time of $J_{4,1}$ on $M_{2,1}$). At this point, there aren't any jobs that require an operation on $M_{1,2}$.
- The jobs are processed on stages 3-17 in the similar fashion. The completion time of all jobs the end of stage 17 along with their weighted tardiness is summarized in Table 5.3.

Table 5.3 shows the summarized initial schedule and weighted tardiness obtained by applying the initial solution finding mechanisms. The weighted tardiness (WT) is evaluated as a job's weight times max [due date – completion time (stage 17), 0]. The total WT is the sum of the weighted tardiness of all jobs. With the initial solution in hand, the effort to find an optimal/near-optimal solution is continued by applying steps of tabu search documented in Section 5.5. The demonstration of the application of tabu search is done via using IS4 (DD/weight) and an optimal/near-optimal solution is finally obtained.

Table 5-3 Initial solutions of example problem

Jobs	EDD		LFJ/ LFM		LWT		DD/Wt ratio		HCR	
	CT	WT	CT	WT	CT	WT	CT	WT	CT	WT
$J_{1,1}$	288	0	501	0	412	0	368	0	368	0
$J_{2,1}$	652	15	454	0	365	0	321	0	321	0
$J_{3,1}$	354	0	716	377	820	0	796	2	796	2
$J_{4,1}$	883	134	815	0	703	0	911	0	911	0
$J_{5,1}$	590	300	385	0	296	354	507	394	507	394
$J_{6,1}$	748	0	766	498	654	324	659	284	659	284
$J_{7,1}$	538	264	646	516	565	274	585	375	585	375
$J_{8,1}$	817	0	880	831	902	0	728	457	728	457
$J_{9,1}$	413	462	575	268	603	396	442	500	442	500
$J_{10,1}$	696	642	252	0	482	481	986	0	986	0
$J_{11,1}$	484	0	833	483	746	897	850	167	850	167
Total	**1817**		**2973**		**2726**		**2179**		**2179**	
CT = Completion time of Job in stage 17, WT = job's weighted tardiness										

Step 1 & 2: All possible interchange (swap) of two jobs are considered. The swap between $J_{1,1}$ and $J_{2,1}$ is ruled out as $J_{2,1}$ cannot be processed on $M_{1,1}$. A similar situation exists for $J_{1,1}$ and $J_{3,1}$. The swap between $J_{1,1}$ and $J_{3,1}$ is ruled out as $J_{3,1}$ cannot be processed on $M_{1,1}$ and $J_{1,1}$ cannot be processed on $M_{3,1}$. The swap between $J_{1,1}$ and $J_{4,1}$ is feasible as both are processed on $M_{1,1}$ and $e_{1,1} < CT\ (J_{4,1}, M_{1,1})$ and $e_{3,1} < CT\ (J_{1,1}, M_{1,1})$. Two new scripts namely *STS* and *STR* have been introduced for comprehensively explaining the search algorithm. *STS* refers to the start of the setup whereas *STR* refers to the start of the run. Swapping $J_{1,1}$ and $J_{4,1}$ results in the following changes on $M_{1,1}$. $STS_r\ (J_{4,1}, M_{1,1}) = 3$, $STR_r\ (J_{4,1}, M_{1,1}) = 22$ and $CT_r\ (J_{4,1}, M_{1,1}) = 56$ whereas $STS_r\ (J_{1,1}, M_{1,1}) = 159$, $STR_r\ (J_{1,1}, M_{1,1}) = 191$ and $CT_r\ (J_{1,1}, M_{1,1}) = 236$. Note the subscript 'r' following *STS*, *STR* and *CT* denotes that they are revised.

The swap between $J_{1,1}$ and $J_{5,1}$ is ruled out as $J_{5,1}$ cannot be processed on $M_{1,1}$. A similar situation exists for $J_{1,1}$ and $J_{6,1}$ as $J_{6,1}$ cannot be processed on $M_{1,1}$. Swap between $J_{1,1}$ and $J_{7,1}$ is also infeasible because $J_{1,1}$ cannot be processed on $M_{3,1}$. The swap between $J_{1,1}$ and $J_{8,1}$ is feasible as both are processed on $M_{1,1}$ and $e_{1,1} < CT$ ($J_{8,1}$, $M_{1,1}$) and $e_{8,1} < CT$ ($J_{1,1}$, $M_{1,1}$). Swapping $J_{1,1}$ and $J_{8,1}$ results in the following changes on $M_{1,1}$. STS_r ($J_{8,1}$, $M_{1,1}$) = 3, STR_r ($J_{8,1}$, $M_{1,1}$) = 18 and CT_r ($J_{8,1}$, $M_{1,1}$) = 55 whereas STS_r ($J_{1,1}$, $M_{1,1}$) = 55, STR_r ($J_{1,1}$, $M_{1,1}$) = 60 and CT_r ($J_{1,1}$, $M_{1,1}$) = 105. Note that according to IS4, $J_{1,1}$ was the first job scheduled on $M_{1,1}$ and $J_{8,1}$ was the second. After swapping $J_{1,1}$ with $J_{8,1}$, $J_{1,1}$ becomes the second job scheduled on $M_{1,1}$ and $J_{8,1}$ becomes the first. Note that the time taken to change over from $J_{8,1}$ to $J_{1,1}$ on $M_{1,1}$ is just 5 units. This information can be located in the setup time matrix provided in the appendix.

The swap between $J_{1,1}$ and $J_{9,1}$ is ruled out as $J_{9,1}$ cannot be processed on $M_{1,1}$. A similar situation exists for $J_{1,1}$ and $J_{10,1}$ as $J_{10,1}$ cannot be processed on $M_{1,1}$. The swap between $J_{1,1}$ and $J_{11,1}$ is feasible as both are processed on $M_{1,1}$ and $e_{1,1} < CT$ ($J_{11,1}$, $M_{1,1}$) and $e_{11,1} < CT$ ($J_{1,1}$, $M_{1,1}$). Swapping $J_{1,1}$ and $J_{11,1}$ results in the following changes on $M_{1,1}$. STS_r ($J_{11,1}$, $M_{1,1}$) = 3, STR_r ($J_{11,1}$, $M_{1,1}$) = 35 and CT_r ($J_{11,1}$, $M_{1,1}$) = 83 whereas STS_r ($J_{1,1}$, $M_{1,1}$) = 127, STR_r ($J_{1,1}$, $M_{1,1}$) =132 and CT_r ($J_{1,1}$, $M_{1,1}$) = 177. The job swapping is continued in the same fashion until all feasible swap moves are made. Table 5.4 shows all feasible swap moves applied to the initial solution along with their TWT value.

Table 5-4 The neighborhood solutions of initial solution as a result of applying swap and insert moves

Swap Moves			
Swap Jobs	**TWT**	**Swap Jobs**	**TWT**
$J_{1,1}$ and $J_{4,1}$	2148	$J_{5,1}$ and $J_{9,1}$	2030
$J_{1,1}$ and $J_{8,1}$	2018	$J_{6,1}$ and $J_{7,1}$	2179
$J_{1,1}$ and $J_{11,1}$	1548	$J_{6,1}$ and $J_{9,1}$	2197
$J_{2,1}$ and $J_{5,1}$	1988	$J_{6,1}$ and $J_{10,1}$	2393

$J_{3,1}$ and $J_{6,1}$	2414	$J_{7,1}$ and $J_{9,1}$	1759
$J_{3,1}$ and $J_{7,1}$	2202	$J_{7,1}$ and $J_{10,1}$	2678
$J_{3,1}$ and $J_{9,1}$	2202	$J_{7,1}$ and $J_{11,1}$	4540
$J_{3,1}$ and $J_{10,1}$	2157	$J_{8,1}$ and $J_{11,1}$	2270
$J_{4,1}$ and $J_{8,1}$	2722	$J_{9,1}$ and $J_{10,1}$	2591
$J_{4,1}$ and $J_{11,1}$	2307		

Insert Moves

Job	**Machine**	**Position**	**TWT**	**Job**	**Machine**	**Position**	**TWT**
$J_{1,1}$	$M_{1,1}$		1309	$J_{7,1}$	$M_{3,1}$		2207
$J_{1,1}$	$M_{1,1}$		1952	$J_{7,1}$	$M_{3,1}$		
$J_{1,1}$	$M_{1,1}$			$J_{9,1}$	$M_{3,1}$		1559
$J_{5,1}$	$M_{2,1}$		1851	$J_{9,1}$	$M_{3,1}$		1590
$J_{5,1}$	$M_{2,1}$		1935	$J_{9,1}$	$M_{3,1}$		
$J_{5,1}$	$M_{2,1}$		1928	$J_{11,1}$	$M_{1,1}$		2042
$J_{5,1}$	$M_{2,1}$		1732	$J_{11,1}$	$M_{1,1}$		1921
$J_{5,1}$	$M_{2,1}$		1654	$J_{11,1}$	$M_{1,1}$		1921
$J_{5,1}$	$M_{2,1}$			$J_{11,1}$	$M_{1,1}$		2222
$J_{7,1}$	$M_{3,1}$		1514	$J_{11,1}$	$M_{1,1}$		2307
$J_{7,1}$	$M_{3,1}$		2371	$J_{11,1}$	$M_{1,1}$		
$J_{7,1}$	$M_{3,1}$		2136				

Insert moves are now considered. $J_{1,1}$ can be inserted to other machines as it can be processed on more than one machine in stage 1 ($J_{1,1}$ can be processed on $M_{1,1}$ and $M_{2,1}$ in stage 1). Inserting $J_{1,1}$ in the first position of $M_{2,1}$ (i.e. preceding $J_{2,1}$) is feasible as $J_{1,1}$

can be processed on $M_{2,1}$ and $e_{1,1} < CT$ ($J_{2,1}$, $M_{2,1}$). The new start and completion times of the jobs scheduled on $M_{2,1}$ are: STS_r ($J_{1,1}$, $M_{2,1}$) = 8, STR_r ($J_{1,1}$, $M_{2,1}$) = 34 and CT_r ($J_{1,1}$, $M_{2,1}$) = 69 whereas : STS_r ($J_{2,1}$, $M_{2,1}$) = 69, STR_r ($J_{2,1}$, $M_{2,1}$) = 82 and CT_r ($J_{2,1}$, $M_{2,1}$) = 116. Note that according to IS4, $J_{2,1}$ was the first job scheduled on $M_{2,1}$ followed by $J_{9,1}$. After inserting $J_{1,1}$ at the first position on $M_{2,1}$, $J_{2,1}$ becomes the second job to be processed on $M_{2,1}$ and $J_{5,1}$ becomes the third job to be processed on $M_{2,1}$.

The next feasible insert move is to insert $J_{1,1}$ to the second position of $M_{2,1}$ (i.e between $J_{2,1}$ and $J_{5,1}$). The new start and completion times of the jobs scheduled on $M_{2,1}$ are STS_r ($J_{1,1}$, $M_{2,1}$) = 59, STR_r ($J_{1,1}$, $M_{2,1}$) = 66 and CT_r ($J_{1,1}$, $M_{2,1}$) = 101 whereas : STS_r ($J_{5,1}$, $M_{2,1}$) = 101, STR_r ($J_{5,1}$, $M_{2,1}$) = 114 and CT_r ($J_{5,1}$, $M_{2,1}$) = 155. The start and completion time of $J_{2,1}$ remains unchanged (i.e. same as explained in the IS2) because $J_{1,1}$ is inserted to the second position on machine $M_{2,1}$.

Inserting $J_{1,1}$ to the third (also the last) position of $M_{2,1}$ (i.e. after $J_{5,1}$) is the next feasible move. The start and completion time of $J_{2,1}$ and $J_{5,1}$ on $M_{2,1}$ remains unaltered. According to IS4, only two jobs were scheduled to be processed on $M_{2,1}$ ($J_{2,1}$ and $J_{5,1}$). Therefore, inserting $J_{1,1}$ to the third position of $M_{2,1}$ doesn't alter the start and completion times of $J_{2,1}$ and $J_{5,1}$. The start and completion times of $J_{1,1}$ scheduled on $M_{2,1}$ (i.e. after being inserted to the last position) is given as follows: STS_r ($J_{1,1}$, $M_{2,1}$) = 147, STR_r ($J_{1,1}$, $M_{2,1}$) = 156 and CT_r ($J_{1,1}$, $M_{2,1}$) = 191. The insert moves are continued in the same fashion for all feasible moves. The overall insert moves applied to IS and their total weighted tardiness values are shown in Table 5.4.

Step 3: The minimum TWT is 1309. The move that results in this value is inserting $J_{1,1}$ at the first position of $M_{2,1}$. The schedule generated by inserting $J_{1,1}$ at the first position of $M_{2,1}$ would be used as the seed for the next iteration. At this point, the following parameters need to be updated:

(1) Tabu List

The primary use of the tabu list is to prevent the search from revisiting previous solutions or repeating the previous moves. As mentioned earlier, whenever a move is made, the tabu list is updated by storing the attributes of the move. For example, if a swap move results in the best solution, then tabu list records the pair of jobs being exchanged and these pairs of jobs are not allowed to exchange positions for the number of iterations indicated by the size of the tabu list unless an aspiration criterion is satisfied. If the best solution is the result of an insert move, then tabu list records the job index along with the position and machine occupied by the job before the move was applied. The job is not allowed to be inserted back to this position of the machine for the number of iterations indicated by the size of the tabu list unless an aspiration criterion is satisfied. In the example problem illustrated above, $J_{1,1}$ is inserted at the first position of $M_{2,1}$ from the first position $M_{1,1}$. Therefore, $J_{1,1}$ is not allowed to be inserted back on the first position of $M_{1,1}$. As mentioned in section 5.5, two types of tabu list sizes are used: fixed tabu list size and variable tabu list size. The tabu list size is evaluated as follows:

- For fixed tabu list size = .04558x – 1.0177 = 4
- For variable tabu list size:
    - Initial = 0.4426x – 0.7869 = 4
    - Decrease = $-0.0254x^2 + 1.1085x - 5.9898 = 3$
    - Increase = $0.0086x^3 - 0.3838x^2 + 6.0924x - 27 = 5$

-(2) Aspiration Level (AL)

The AL is initially set equal to the TWT of the initial solution, which is 2179. Since inserting $J_{1,1}$ to the first position of $M_{2,1}$ yields a TWT of 1309, the AL is updated to be equal to 1309. If a tabu move in the next iteration results in a TWT that is less than 1309, the move is released from its tabu restriction.

(3) Candidate List (CL) and Index List (IL)

Initially, the initial solution (S_0) is admitted to both CL and IL as it is considered as a local optimum. As the solution obtained by inserting $J_{1,1}$ to the first position of $M_{2,1}$ (i.e. S_1) is selected as the best solution, S_1 is admitted into the CL. Since S_1 is better than S_0, S_1 receives a star, which means that is has the potential to become a local optimum. At this point, the CL has two entries and IL has only one entry:

CL: { [$J_{1,1}/M_{1,1}$, $J_{2,1}/M_{2,1}$, $J_{1,1}/M_{1,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{2,1}$, $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{3,1}$,

$J_{8,1}/M_{1,1}$, $J_{9,1}/M_{3,1}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{1,1}$]

[$\boldsymbol{J_{1,1}/M_{2,1}}$, $J_{2,1}/M_{2,1}$, $J_{1,1}/M_{1,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{2,1}$, $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{3,1}$,

$J_{8,1}/M_{1,1}$, $J_{9,1}/M_{3,1}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{1,1}$] }

IL: { [$J_{1,1}/M_{2,1}$, $J_{2,1}/M_{2,1}$, $J_{1,1}/M_{1,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{2,1}$, $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{3,1}$,

$J_{8,1}/M_{1,1}$, $J_{9,1}/M_{3,1}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{1,1}$]

(4) Number of iterations without improvement (IWOI)

Initially, IWOI equals to zero. Since there is an improvement in the TWT, i.e. a change from 2179 to 1309, the IWOI remains to be zero.

(5) Long-term memory (LTM) Matrix

As mentioned in Section 5.5, the LTM matrix records the tally of the jobs processed on the machines. For this research, we consider LTM matrix consisting of 11 × 3 cells. Note that only first stage has multiple machines (three to be precise). Stages 2-17 have only one machine. The tally for jobs corresponding to machines in stages 2-17 will always be maximum (since there is only one machine per stage). It is meaningless to fix these jobs to the machine since even without doing so the jobs will not be processed on other machines throughout the course of the search process. Thus for the purpose of this research, we are primarily interested in the 3 machines that belong to stage 1. In this case, we consider the LTM matrix consisting of 11 jobs and 3 machines (11 × 3 cells). The first iteration obtained by inserting $J_{1,1}$ at the first position of $M_{2,1}$ results in the following entries in LTM matrix, as presented in Table 5.5.

Step 4: To terminate the search, two stopping criterions are used: IWOI max and IL max. For fixed and variable tabu list, the stopping criteria are evaluated as follows:

- IWOI = $-0.0141x^2 + .6741x – 1.8444 = 4$
- IL = $0.0109x^3 – 0.5065x^2 + 7.8506x – 35.571 = 4$

The search is terminated if IWOI max reaches 4 or ILmax reaches 4, whichever comes first.

Table 5-5 Entries into the LTM matrix after perturbing the initial solution

Job Index	$M_{1,1}$	$M_{2,1}$	$M_{3,1}$
$J_{1,1}$	0	1	-
$J_{2,1}$	-	1	-
$J_{3,1}$	-	-	1
$J_{4,1}$	1	-	-
$J_{5,1}$	-	1	0
$J_{6,1}$	-	-	1
$J_{7,1}$	0	-	1
$J_{8,1}$	1	-	-
$J_{9,1}$	-	0	1
$J_{10,1}$	-	-	1
$J_{11,1}$	1	-	0

- For variable tabu list size:

The ITmax and IWOImax are evaluated the same way as in fixed tabu list size. The ILmax is used in conjunction with the following steps:

(i) If there is no improvement in the last $\lceil 4/3 \rceil = 1$ iterations with the initial size of tabu list, decrease the size of tabu list to the decreased size evaluated in step 3.

(ii) If there is no improvement in the last $\lceil 4/3 \rceil = 1$ iterations with the decreased size of tabu list, increase the size of tabu list to the increased size evaluated in step 3.

(iii) If there is no improvement in the last $\lceil 4/3 \rceil = 1$ iterations with the increased size of tabu list, terminate the search.

At this point of the search, both stopping criteria are not met. Thus, the search is continued until one of the stopping criteria is met. In this example, the search is terminated after 14 iterations. The stopping criterion activated to terminate the search is ILmax (i.e. when the number of entries into the IL has reached 4). The results of search using the fixed size of tabu list and short-term memory are summarized in Table 5.6.

Table 5-6 Results of tabu search applied to the initial solution of the example problem

Iteration No.	Move applied	Entry into the CL	TWT	Entry into the IL
0	--	[$J_{1,1}/M_{1,1}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{2,1}$, $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{3,1}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{3,1}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{1,1}$]**	2179	2179
1	Insert ($J_{1,1}$,$M_{2,1}$,P1)	**[**$\boldsymbol{J_{1,1}/M_{2,1}}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{2,1}$, $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{3,1}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{3,1}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{1,1}$]*	1309	
2	Swap ($J_{1,1}$, $J_{5,1}$)	**[**$\boldsymbol{J_{1,1}/M_{2,1}}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $\boldsymbol{J_{5,1}/M_{2,1}}$, $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{3,1}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{3,1}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{1,1}$]*	923	
3	Insert ($J_{9,1}$,$M_{2,1}$,P1)	[$J_{1,1}/M_{2,1}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{2,1}$, $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{3,1}$, $J_{8,1}/M_{1,1}$, $\boldsymbol{J_{9,1}/M_{2,1}}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{1,1}$]*	864	
4	Insert ($J_{1,1}$,$M_{1,1}$,P3)	**[**$\boldsymbol{J_{1,1}/M_{1,1}}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{2,1}$, $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{3,1}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{2,1}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{1,1}$]**	799	799
5	Swap ($J_{1,1}$, $J_{4,1}$)	**[**$\boldsymbol{J_{1,1}/M_{1,1}}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $\boldsymbol{J_{4,1}/M_{1,1}}$, $J_{5,1}/M_{2,1}$, $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{3,1}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{2,1}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{1,1}$]	864	

6	Insert ($J_{7,1}$,$M_{1,1}$,P1)	$[J_{1,1}/M_{1,1}, J_{2,1}/M_{2,1}, J_{3,1}/M_{3,1}, J_{4,1}/M_{1,1}, J_{5,1}/M_{2,1}, J_{6,1}/M_{3,1}, \boldsymbol{J_{7,1}/M_{1,1}}, J_{8,1}/M_{1,1}, J_{9,1}/M_{2,1}, J_{10,1}/M_{3,1}, J_{11,1}/M_{1,1}]$	878	
7	Insert ($J_{11,1}$,$M_{3,1}$,P3)	$[J_{1,1}/M_{1,1}, J_{2,1}/M_{2,1}, J_{3,1}/M_{3,1}, J_{4,1}/M_{1,1}, J_{5,1}/M_{2,1}, J_{6,1}/M_{3,1}, J_{7,1}/M_{1,1}, J_{8,1}/M_{1,1}, J_{9,1}/M_{2,1}, J_{10,1}/M_{3,1}, \boldsymbol{J_{11,1}/M_{3,1}}]$	885	
8	Insert ($J_{5,1}$,$M_{3,1}$,P4)	$[J_{1,1}/M_{1,1}, J_{2,1}/M_{2,1}, J_{3,1}/M_{3,1}, J_{4,1}/M_{1,1}, \boldsymbol{J_{5,1}/M_{3,1}}, J_{6,1}/M_{3,1}, J_{7,1}/M_{1,1}, J_{8,1}/M_{1,1}, J_{9,1}/M_{2,1}, J_{10,1}/M_{3,1}, J_{11,1}/M_{3,1}]^*$	665	
9	Swap ($J_{5,1}$,$J_{10,1}$)	$[J_{1,1}/M_{1,1}, J_{2,1}/M_{2,1}, J_{3,1}/M_{3,1}, J_{4,1}/M_{1,1}, \boldsymbol{J_{5,1}/M_{3,1}}, J_{6,1}/M_{3,1}, J_{7,1}/M_{1,1}, J_{8,1}/M_{1,1}, J_{9,1}/M_{2,1}, \boldsymbol{J_{10,1}/M_{3,1}}, J_{11,1}/M_{3,1}]^*$	580	
10	Swap ($J_{6,1}$,$J_{3,1}$)	$[J_{1,1}/M_{1,1}, J_{2,1}/M_{2,1}, \boldsymbol{J_{3,1}/M_{3,1}}, J_{4,1}/M_{1,1}, J_{5,1}/M_{3,1}, \boldsymbol{J_{6,1}/M_{3,1}}, J_{7,1}/M_{1,1}, J_{8,1}/M_{1,1}, J_{9,1}/M_{2,1}, J_{10,1}/M_{3,1}, J_{11,1}/M_{3,1}]^{**}$	400	400
11	Swap ($J_{4,1}$,$J_{1,1}$)	$[\boldsymbol{J_{1,1}/M_{1,1}}, J_{2,1}/M_{2,1}, J_{3,1}/M_{3,1}, \boldsymbol{J_{4,1}/M_{1,1}}, J_{5,1}/M_{3,1}, J_{6,1}/M_{3,1}, J_{7,1}/M_{1,1}, J_{8,1}/M_{1,1}, J_{9,1}/M_{2,1}, J_{10,1}/M_{3,1}, J_{11,1}/M_{3,1}]$	436	
12	Insert ($J_{9,1}$,$M_{3,1}$,P2)	$[J_{1,1}/M_{1,1}, J_{2,1}/M_{2,1}, J_{3,1}/M_{3,1}, J_{4,1}/M_{1,1}, J_{5,1}/M_{3,1}, J_{6,1}/M_{3,1}, J_{7,1}/M_{1,1}, J_{8,1}/M_{1,1}, \boldsymbol{J_{9,1}/M_{3,1}}, J_{10,1}/M_{3,1}, J_{11,1}/M_{3,1}]$	496	
13	Insert ($J_{11,1}$,$M_{1,1}$,P3)	$[J_{1,1}/M_{1,1}, J_{2,1}/M_{2,1}, J_{3,1}/M_{3,1}, J_{4,1}/M_{1,1}, J_{5,1}/M_{3,1}, J_{6,1}/M_{3,1}, J_{7,1}/M_{1,1}, J_{8,1}/M_{1,1}, J_{9,1}/M_{3,1}, J_{10,1}/M_{3,1}, \boldsymbol{J_{11,1}/M_{1,1}}]^{**}$	329	329
14	Swap ($J_{4,1}$,$J_{1,1}$)	$[J_{1,1}/M_{1,1}, J_{2,1}/M_{2,1}, J_{3,1}/M_{3,1}, \boldsymbol{J_{4,1}/M_{1,1}}, J_{5,1}/M_{3,1}, J_{6,1}/M_{3,1}, J_{7,1}/M_{1,1}, J_{8,1}/M_{1,1}, J_{9,1}/M_{3,1}, J_{10,1}/M_{3,1}, \boldsymbol{J_{11,1}/M_{1,1}}]$	506	

The CL has 15 entries and the IL has 4 entries. The best solution obtained by short term memory function is found at the 13th iteration with a TWT value of 329. The best solution is pointing to the following schedule: [$J_{1,1}/M_{1,1}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{3,1}$, $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{1,1}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{3,1}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{1,1}$**].**

Step 5: At this point, the search can be restarted from a different region of the solution space. The restarting point is defined from the LTM matrix. The entries into the LTM matrix at the time the search is terminated is shown in Table 5.7. For the maximum frequency approach, the cells that have the maximum tally, is chosen. Recall that for maximum frequency approach, we only consider jobs that can be processed on multiple machines. Among jobs that can be processed on more than one machine, the cell corresponding to $J_{1,1}$ and $M_{1,1}$ has the highest tally. Thus, the first restart solution based on maximal frequency is generated by fixing $J_{1,1}$ on $M_{1,1}$. The first restart solution is [$J_{1,1}/M_{1,1}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{3,1}$, $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{1,1}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{3,1}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{1,1}$]. The tabu list and IWOI are re-initialized back to zero. The AL is reset to the TWT of the restart solution, which is equal to 2179. Repeat Step 1 to Step 4 using the first restart solution as a new starting point.

Table 5-7 Entries into the LTM matrix at the end of the search using the initial solution

Job Index	$M_{1,1}$	$M_{2,1}$	$M_{3,1}$
$J_{1,1}$	11	3	-
$J_{2,1}$	-	14	-
$J_{3,1}$	-	-	14
$J_{4,1}$	14	-	-
$J_{5,1}$	-	7	7
$J_{6,1}$	-	-	14
$J_{7,1}$	9	-	5
$J_{8,1}$	14	-	-
$J_{9,1}$	-	9	5
$J_{10,1}$	-	-	14
$J_{11,1}$	8	-	6

Based on the LTM-max, the results obtained with the first restart are shown in Table 5.8. The underlined job indicates that it is fixed to the machine throughout the first restart. The first restart is terminated after 10 iterations because the entries into the IL reached its maximum (4). The best solution obtained from the first LTM-max restart is found at the sixth iteration with a TWT value of 599.

Table 5-8 Results from the first restart based on maximal frequency

Iteration No.	Move applied	Entry into the CL	TWT	Entry into the IL
0	--	$[J_{1,1}/M_{1,1}, J_{2,1}/M_{2,1}, J_{3,1}/M_{3,1}, J_{4,1}/M_{1,1}, J_{5,1}/M_{2,1}, J_{6,1}/M_{3,1}, J_{7,1}/M_{3,1}, J_{8,1}/M_{1,1}, J_{9,1}/M_{3,1}, J_{10,1}/M_{3,1}, J_{11,1}/M_{1,1}]$**	2179	2179
1	Insert ($J_{7,1}$,$M_{1,1}$,P1)	$[\underline{J_{1,1}/M_{1,1}}, J_{2,1}/M_{2,1}, J_{3,1}/M_{3,1}, J_{4,1}/M_{1,1}, J_{5,1}/M_{2,1}, J_{6,1}/M_{3,1}, \boldsymbol{J_{7,1}/M_{1,1}}, J_{8,1}/M_{1,1}, J_{9,1}/M_{3,1}, J_{10,1}/M_{3,1}, J_{11,1}/M_{1,1}]$*	1514	
2	Insert ($J_{11,1}$,$M_{3,1}$,P4)	$[\underline{J_{1,1}/M_{1,1}}, J_{2,1}/M_{2,1}, J_{3,1}/M_{3,1}, J_{4,1}/M_{1,1}, J_{5,1}/M_{2,1}, J_{6,1}/M_{3,1}, J_{7,1}/M_{1,1}, J_{8,1}/M_{1,1}, J_{9,1}/M_{3,1}, J_{10,1}/M_{3,1}, \boldsymbol{J_{11,1}/M_{3,1}}]$**	1150	1150
3	Swap ($J_{3,1}$, $J_{6,1}$)	$[\underline{J_{1,1}/M_{1,1}}, J_{2,1}/M_{2,1}, \boldsymbol{J_{3,1}/M_{3,1}}, J_{4,1}/M_{1,1}, J_{5,1}/M_{2,1}, \boldsymbol{J_{6,1}/M_{3,1}}, J_{7,1}/M_{1,1}, J_{8,1}/M_{1,1}, J_{9,1}/M_{3,1}, J_{10,1}/M_{3,1}, J_{11,1}/M_{3,1}]$	1178	
4	Swap ($J_{3,1}$, $J_{9,1}$)	$[\underline{J_{1,1}/M_{1,1}}, J_{2,1}/M_{2,1}, \boldsymbol{J_{3,1}/M_{3,1}}, J_{4,1}/M_{1,1}, J_{5,1}/M_{2,1}, J_{6,1}/M_{3,1}, J_{7,1}/M_{1,1}, J_{8,1}/M_{1,1}, \boldsymbol{J_{9,1}/M_{3,1}}, J_{10,1}/M_{3,1}, J_{11,1}/M_{3,1}]$*	1011	
5	Insert ($J_{5,1}$,$M_{3,1}$,P5)	$[\underline{J_{1,1}/M_{1,1}}, J_{2,1}/M_{2,1}, J_{3,1}/M_{3,1}, J_{4,1}/M_{1,1}, \boldsymbol{J_{5,1}/\mathrm{M}_{3,1}}, J_{6,1}/M_{3,1}, J_{7,1}/M_{1,1}, J_{8,1}/M_{1,1}, J_{9,1}/M_{3,1}, J_{10,1}/M_{3,1}, J_{11,1}/M_{3,1}]$*	666	
6	Swap ($J_{5,1}$, $J_{10,1}$)	$[\underline{J_{1,1}/M_{1,1}}, J_{2,1}/M_{2,1}, J_{3,1}/M_{3,1}, J_{4,1}/M_{1,1}, \boldsymbol{J_{5,1}/M_{3,1}}, J_{6,1}/M_{3,1}, J_{7,1}/M_{1,1}, J_{8,1}/M_{1,1}, J_{9,1}/M_{3,1}, \boldsymbol{J_{10,1}/M_{3,1}}, J_{11,1}/M_{3,1}]$**	599	599
7	Swap ($J_{10,1}$, $J_{11,1}$)	$[\underline{J_{1,1}/M_{1,1}}, J_{2,1}/M_{2,1}, J_{3,1}/M_{3,1}, J_{4,1}/M_{1,1}, J_{5,1}/M_{3,1}, J_{6,1}/M_{3,1}, J_{7,1}/M_{1,1}, J_{8,1}/M_{1,1}, J_{9,1}/M_{3,1}, \boldsymbol{J_{10,1}/M_{3,1}}, \boldsymbol{J_{11,1}/M_{3,1}}]$	881	
8	Insert ($J_{7,1}$,$M_{3,1}$,P2)	$[\underline{J_{1,1}/M_{1,1}}, J_{2,1}/M_{2,1}, J_{3,1}/M_{3,1}, J_{4,1}/M_{1,1}, J_{5,1}/M_{3,1}, J_{6,1}/M_{3,1}, \boldsymbol{J_{7,1}/M_{3,1}}, J_{8,1}/M_{1,1}, J_{9,1}/M_{3,1}, J_{10,1}/M_{3,1}, J_{11,1}/M_{3,1}]$	957	

9	Swap ($J_{5,1}$,$J_{11,1}$)	[$J_{1,1}/M_{1,1}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $\boldsymbol{J_{5,1}/M_{3,1}}$, $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{3,1}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{3,1}$, $J_{10,1}/M_{3,1}$, $\boldsymbol{J_{11,1}/M_{3,1}}$]**	927	927
10	Swap ($J_{6,1}$,$J_{3,1}$)	[$J_{1,1}/M_{1,1}$, $J_{2,1}/M_{2,1}$, $\boldsymbol{J_{3,1}/M_{3,1}}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{3,1}$, $\boldsymbol{J_{6,1}/M_{3,1}}$, $J_{7,1}/M_{3,1}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{3,1}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{3,1}$]	1104	

Since the total number of restarts is set equal to 2, the search process is poised to begin its second restart. Again, the restarting point is determined by selecting the job-machine pair with maximum frequency from the LTM matrix. The entries into the LTM matrix at the termination of the first restart are shown in Table 5.9. Using the row-wise first best strategy, $J_{7,1}$ is fixed on machine $M_{1,1}$ because it has the maximum frequency. The seed for the second restart is obtained by fixing $J_{7,1}$ on machine $M_{1,1}$ in the initial solution. Note that in the initial solution, $J_{7,1}$ is scheduled to be processed on machine $M_{3,1}$. Therefore, $J_{7,1}$ is removed from $M_{3,1}$ and inserted at the first position of $M_{1,1}$. The tabu list and IWOI are re-initialized back to zero. Repeat Step 1 to Step 4 using the second restart solution as a new starting point.

Table 5-9 Entries into the LTM matrix at the end of the first restart based on maximum frequency

Job Index	$\boldsymbol{M_{1,1}}$	$\boldsymbol{M_{2,1}}$	$\boldsymbol{M_{3,1}}$
$\boldsymbol{J_{1,1}}$	21	3	-
$\boldsymbol{J_{2,1}}$	-	24	-
$\boldsymbol{J_{3,1}}$	-	-	24
$\boldsymbol{J_{4,1}}$	24	-	-
$\boldsymbol{J_{5,1}}$	-	11	13
$\boldsymbol{J_{6,1}}$	-	-	24
$\boldsymbol{J_{7,1}}$	16	-	8
$\boldsymbol{J_{8,1}}$	24	-	-

$\boldsymbol{J_{9,1}}$	-	9	15
$\boldsymbol{J_{10,1}}$	-	-	24
$\boldsymbol{J_{11,1}}$	9	-	15

The results obtained with the second restart based on maximum frequency are shown in Table 5.10. The underlined job indicates that it is fixed to the machine throughout the second restart. The second restart is terminated after nine iterations when IL entries reach its maximum (4).

Table 5-10 Results from the second restart based on maximal frequency

Iteration No.	Move applied	Entry into the CL	TWT	Entry into the IL
0	--	$[J_{1,1}/M_{1,1}, J_{2,1}/M_{2,1}, J_{3,1}/M_{3,1}, J_{4,1}/M_{1,1}, J_{5,1}/M_{2,1}, J_{6,1}/M_{1,1}, \underline{J_{7,1}/M_{3,1}}, J_{8,1}/M_{1,1}, J_{9,1}/M_{3,1}, J_{10,1}/M_{3,1}, J_{11,1}/M_{1,1}]^{**}$	1775	1775
1	Swap $(J_{1,1}, J_{8,1})$	$[\mathbf{J_{1,1}/M_{1,1}}, J_{2,1}/M_{2,1}, J_{3,1}/M_{3,1}, J_{4,1}/M_{1,1}, J_{5,1}/M_{2,1}, J_{6,1}/M_{1,1}, \underline{J_{7,1}/M_{3,1}}, \mathbf{J_{8,1}/M_{1,1}}, J_{9,1}/M_{3,1}, J_{10,1}/M_{3,1}, J_{11,1}/M_{1,1}]^{*}$	1012	
2	Insert $(J_{5,1},M_{3,1},P4)$	$[J_{1,1}/M_{1,1}, J_{2,1}/M_{2,1}, J_{3,1}/M_{3,1}, J_{4,1}/M_{1,1}, \mathbf{J_{5,1}/M_{3,1}}, J_{6,1}/M_{1,1}, \underline{J_{7,1}/M_{3,1}}, J_{8,1}/M_{1,1}, J_{9,1}/M_{3,1}, J_{10,1}/M_{3,1}, J_{11,1}/M_{1,1}]^{**}$	653	653
3	Swap $(J_{1,1}, J_{11,1})$	$[\mathbf{J_{1,1}/M_{1,1}}, J_{2,1}/M_{2,1}, J_{3,1}/M_{3,1}, J_{4,1}/M_{1,1}, J_{5,1}/M_{3,1}, J_{6,1}/M_{1,1}, \underline{J_{7,1}/M_{3,1}}, J_{8,1}/M_{1,1}, J_{9,1}/M_{3,1}, J_{10,1}/M_{3,1}, \mathbf{J_{11,1}/M_{1,1}}]$	714	
4	Swap $(J_{5,1}, J_{10,1})$	$[J_{1,1}/M_{1,1}, J_{2,1}/M_{2,1}, J_{3,1}/M_{3,1}, J_{4,1}/M_{1,1}, \mathbf{J_{5,1}/M_{3,1}}, J_{6,1}/M_{1,1}, \underline{J_{7,1}/M_{3,1}}, J_{8,1}/M_{1,1}, J_{9,1}/M_{3,1}, \mathbf{J_{10,1}/M_{3,1}}, J_{11,1}/M_{1,1}]$	714	
5	Swap $(J_{1,1}, J_{4,1})$	$[\mathbf{J_{1,1}/M_{1,1}}, J_{2,1}/M_{2,1}, J_{3,1}/M_{3,1}, \mathbf{J_{4,1}/M_{1,1}}, J_{5,1}/M_{3,1}, J_{6,1}/M_{1,1}, \underline{J_{7,1}/M_{3,1}}, J_{8,1}/M_{1,1}, J_{9,1}/M_{3,1}, J_{10,1}/M_{3,1}, J_{11,1}/M_{1,1}]^{*}$	587	
6	Insert $(J_{9,1},M_{2,1},P1)$	$[J_{1,1}/M_{1,1}, J_{2,1}/M_{2,1}, J_{3,1}/M_{3,1}, J_{4,1}/M_{1,1}, J_{5,1}/M_{3,1}, J_{6,1}/M_{1,1}, \underline{J_{7,1}/M_{3,1}}, J_{8,1}/M_{1,1},$	533	533

		$\mathbf{J_{9,1}/M_{1,1}}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{1,1}$]**		
7	Insert ($J_{11,1}$,$M_{3,1}$,P3)	[$J_{1,1}/M_{1,1}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{3,1}$, $J_{6,1}/M_{1,1}$, $\underline{J_{7,1}/M_{3,1}}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{1,1}$, $J_{10,1}/M_{3,1}$, $\mathbf{J_{11,1}/M_{3,1}}$]	580	
8	Swap ($J_{3,1}$,$J_{6,1}$)	[$J_{1,1}/M_{1,1}$, $J_{2,1}/M_{2,1}$, $\mathbf{J_{3,1}/M_{3,1}}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{3,1}$, $\mathbf{J_{6,1}/M_{1,1}}$, $\underline{J_{7,1}/M_{3,1}}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{1,1}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{3,1}$]**	400	400
9	Insert ($J_{9,1}$,$M_{3,1}$,P2)	[$J_{1,1}/M_{1,1}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{3,1}$, $J_{6,1}/M_{1,1}$, $\underline{J_{7,1}/M_{3,1}}$, $J_{8,1}/M_{1,1}$, $\mathbf{J_{9,1}/M_{3,1}}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{3,1}$]	451	

Table 5.11 summarizes the best solutions obtained from the initial solution and the two restarts using LTM-max. The table shows that the best solutions obtained by two restarts are not any better than the best solution obtained by the initial search. The quality of the best solutions obtained in the two restarts is actually much inferior than the one obtained in the initial search. For this particular problem, the long term memory did not improve the quality of solution obtained by the short term memory. There are two possible reasons for this. First, the best solution obtained in the initial search is the optimal solution. Hence the long term memory was unable to identify a better solution. Second, the approach used in the long-term memory function, although is capable of directing the search to a different region, is unable to identify a better solution in the new region than the one already found with short term memory. We further explore the search space by applying the long term memory function.

Table 5-11 Summary of results for the entire search process based on LTM-max

Restart Number	Best solutions obtained	TWT
Initial	[$J_{1,1}/M_{1,1}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{3,1}$, $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{1,1}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{3,1}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{1,1}$]	329
First	[$\underline{\mathbf{J_{1,1}/M_{1,1}}}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{3,1}$, $J_{6,1}/M_{3,1}$,	599

	$J_{7,1}/M_{1,1}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{3,1}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{3,1}$]	
Second	[$J_{1,1}/M_{1,1}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{3,1}$, $J_{6,1}/M_{1,1}$, **$\underline{J_{7,1}/M_{3,1}}$** , $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{1,1}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{3,1}$]	400

Referring to the LTM matrix at the time of termination of the initial search in Table 5.7, the job-machine pair with minimum frequency, which is 3, would be $J_{1,1}$ on $M_{2,1}$. Therefore, the starting point for the first restart using the minimum frequency will be generated from the initial solution by fixing $J_{1,1}$ on $M_{2,1}$. In the initial solution, $J_{1,1}$ is processed on $M_{1,1}$. $J_{1,1}$ is removed from $M_{1,1}$ and is inserted into the first position of $M_{2,1}$. This insert move will cause changes in start and completion times of the jobs processed on $M_{1,1}$ and $M_{2,1}$. The starting point for the first restart using minimum frequency is [$\underline{J_{1,1}/M_{2,1}}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{2,1}$, $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{3,1}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{3,1}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{1,1}$] with a TWT value of 1309. Using this solution from LTM-min, the results obtained with the first restart are shown in Table 5.12. $J_{1,1}$ on $M_{2,1}$ is underlined as a sign that $J_{1,1}$ is fixed to $M_{2,1}$ throughout the first restart. The first restart is terminated after seven iterations because entries into the index list reached their maximum (4). The best solution from the first restart is obtained at sixth iteration with a TWT of 1052.

Table 5-12 Results of first restart based on minimum frequency

Iteration No.	Move applied	Entry into the CL	TWT	Entry into the IL
0	--	[$\underline{J_{1,1}/M_{2,1}}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{2,1}$, $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{3,1}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{3,1}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{1,1}$]**	1309	1309
1	Insert ($J_{5,1}$,$M_{3,1}$,P5)	[$\underline{J_{1,1}/M_{2,1}}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, **$J_{5,1}/M_{3,1}$** , $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{3,1}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{3,1}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{1,1}$]**	1072	1072
2	Swap ($J_{5,1}$, $J_{10,1}$)	[$\underline{J_{1,1}/M_{2,1}}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, **$J_{5,1}/M_{3,1}$** , $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{3,1}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{3,1}$, **$J_{10,1}/M_{3,1}$** , $J_{11,1}/M_{1,1}$]	1072	
	Swap	[$\underline{J_{1,1}/M_{2,1}}$, $J_{2,1}/M_{2,1}$, **$J_{3,1}/M_{3,1}$**, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{3,1}$, $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{3,1}$, $J_{8,1}/M_{1,1}$,		

3	($J_{3,1}$, $J_{10,1}$)	$J_{9,1}/M_{3,1}$, $\mathbf{J_{10,1}/M_{3,1}}$, $J_{11,1}/M_{1,1}$]	1099	
4	Swap ($J_{7,1}$, $J_{11,1}$)	[$J_{1,1}/M_{2,1}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{3,1}$, $J_{6,1}/M_{3,1}$, $\mathbf{J_{7,1}/M_{1,1}}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{3,1}$, $J_{10,1}/M_{3,1}$, $\mathbf{J_{11,1}/M_{3,1}}$]**	1063	1063
5	Swap ($J_{7,1}$, $J_{8,1}$)	[$J_{1,1}/M_{2,1}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{3,1}$, $J_{6,1}/M_{3,1}$, $\mathbf{J_{7,1}/M_{1,1}}$, $\mathbf{J_{8,1}/M_{1,1}}$, $J_{9,1}/M_{3,1}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{3,1}$]	1075	
6	Swap ($J_{9,1}$, $J_{11,1}$)	[$J_{1,1}/M_{2,1}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{3,1}$, $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{1,1}$, $J_{8,1}/M_{1,1}$, $\mathbf{J_{9,1}/M_{3,1}}$, $J_{10,1}/M_{3,1}$, $\mathbf{J_{11,1}/M_{3,1}}$]**	1052	1052
7	Swap ($J_{3,1}$, $J_{5,1}$)	[$J_{1,1}/M_{2,1}$, $J_{2,1}/M_{2,1}$, $\mathbf{J_{3,1}/M_{3,1}}$, $J_{4,1}/M_{1,1}$, $\mathbf{J_{5,1}/M_{3,1}}$, $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{1,1}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{3,1}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{3,1}$]	1088	

The second restart based on LTM-min would use the information provided by the LTM matrix at the time of termination of the first restart. This matrix is shown in Table 5.13. The minimum frequency, which is 7, is pointing to $J_{5,1}$ on $M_{2,1}$. Thus $J_{5,1}$ would be fixed on $M_{2,1}$ throughout the second restart. Since $J_{5,1}$ is processed on $M_{2,1}$ in the initial solution, the starting point for the second restart will be generated from the initial solution itself. Using the initial solution, the search is restarted in the similar fashion as in LTM-max and the results are shown in Table 5.14.

Table 5-13 Entries into the LTM matix at the end of first restart based on minimum frequency

Job Index	$\mathbf{M_{1,1}}$	$\mathbf{M_{2,1}}$	$\mathbf{M_{3,1}}$
$\mathbf{J_{1,1}}$	11	10	-
$\mathbf{J_{2,1}}$	-	21	-
$\mathbf{J_{3,1}}$	-	-	21

$J_{4,1}$	21	-	-
$J_{5,1}$	-	7	14
$J_{6,1}$	-	-	21
$J_{7,1}$	13	-	8
$J_{8,1}$	21	-	-
$J_{9,1}$	-	9	12
$J_{10,1}$	-	-	21
$J_{11,1}$	11	-	10

The summary of the best solutions obtained from the initial search and the two restarts using LTM-min is shown in Table 5.15. The first restart using LTM-min yields a solution that is inferior to the one obtained by the initial search. Therefore the solution obtained by the initial search should be optimal/near optimal (since intensification and diversification yield inferior solutions). In chapter 6, the optimal solution for this problem is found using optimization software (CPLEX) and the results are discussed for various small problem instances.

Table 5-14 Results of second restart based on minimum frequency

Iteration No.	Move applied	Entry into the CL	TWT	Entry into the IL
0	--	[$J_{1,1}/M_{1,1}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $\underline{J_{5,1}/M_{2,1}}$, $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{3,1}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{3,1}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{1,1}$]**	2179	2179
1	Insert ($J_{1,1}$,$M_{2,1}$,P1)	[**$J_{1,1}/M_{2,1}$**, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $\underline{J_{5,1}/M_{2,1}}$, $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{3,1}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{3,1}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{1,1}$]*	1309	

2	Swap ($J_{7,1}$,$J_{11,1}$)	$J_{1,1}/M_{2,1}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{2,1}$, $J_{6,1}/M_{3,1}$, **$J_{7,1}/M_{1,1}$** , $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{3,1}$, $J_{10,1}/M_{3,1}$, **$J_{11,1}/M_{3,1}$**]*	1237	
3	Swap ($J_{3,1}$,$J_{10,1}$)	[$J_{1,1}/M_{2,1}$, $J_{2,1}/M_{2,1}$, **$J_{3,1}/M_{3,1}$**, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{2,1}$, $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{1,1}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{3,1}$, **$J_{10,1}/M_{3,1}$** , $J_{11,1}/M_{3,1}$]**	1181	1181
4	Swap ($J_{6,1}$,$J_{11,1}$)	[$J_{1,1}/M_{2,1}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{2,1}$, **$J_{6,1}/M_{3,1}$** , $J_{7,1}/M_{1,1}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{3,1}$, $J_{10,1}/M_{3,1}$, **$J_{11,1}/M_{3,1}$**]	1337	
5	Swap ($J_{7,1}$,$J_{8,1}$)	[$J_{1,1}/M_{2,1}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{2,1}$, $J_{6,1}/M_{3,1}$, **$J_{7,1}/M_{1,1}$, $J_{8,1}/M_{1,1}$** , $J_{9,1}/M_{3,1}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{3,1}$]*	1267	
6	Insert ($J_{1,1}$,$M_{1,1}$,P3)	[**$J_{1,1}/M_{1,1}$**, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{2,1}$, $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{1,1}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{3,1}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{3,1}$]*	1173	
7	Swap ($J_{1,1}$,$J_{4,1}$)	[**$J_{1,1}/M_{1,1}$**, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, **$J_{4,1}/M_{1,1}$** , $J_{5,1}/M_{2,1}$, $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{1,1}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{3,1}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{3,1}$]*	914	
8	Insert ($J_{9,1}$,$M_{2,1}$,P1)	[$J_{1,1}/M_{1,1}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{2,1}$, $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{1,1}$, $J_{8,1}/M_{1,1}$, **$J_{9,1}/M_{2,1}$** , $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{3,1}$]**	909	909
9	Swap ($J_{2,1}$,$J_{9,1}$)	[$J_{1,1}/M_{1,1}$, **$J_{2,1}/M_{2,1}$**, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{2,1}$, $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{1,1}$, $J_{8,1}/M_{1,1}$, **$J_{9,1}/M_{2,1}$** , $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{3,1}$]	921	
10	Swap ($J_{3,1}$,$J_{10,1}$)	[$J_{1,1}/M_{1,1}$, $J_{2,1}/M_{2,1}$, **$J_{3,1}/M_{3,1}$**, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{2,1}$, $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{1,1}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{2,1}$, **$J_{10,1}/M_{3,1}$** , $J_{11,1}/M_{3,1}$]	979	
11	Swap ($J_{3,1}$,$J_{6,1}$)	[$J_{1,1}/M_{1,1}$, $J_{2,1}/M_{2,1}$, **$J_{3,1}/M_{3,1}$**, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{2,1}$, **$J_{6,1}/M_{3,1}$** , $J_{7,1}/M_{1,1}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{2,1}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{3,1}$]*	833	
12	Swap ($J_{6,1}$,$J_{10,1}$)	[$J_{1,1}/M_{1,1}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{2,1}$, **$J_{6,1}/M_{3,1}$** , $J_{7,1}/M_{1,1}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{2,1}$, **$J_{10,1}/M_{3,1}$** , $J_{11,1}/M_{3,1}$]**	579	579
13	Swap ($J_{10,1}$,$J_{11,1}$)	[$J_{1,1}/M_{1,1}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{2,1}$, $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{1,1}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{2,1}$, **$J_{10,1}/M_{3,1}$** , **$J_{11,1}/M_{3,1}$**]	891	

Table 5-15 Summary of results for the entire search process based on LTM-min

Restart Number	Best solutions obtained	TWT
Initial	[$J_{1,1}/M_{1,1}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{3,1}$, $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{1,1}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{3,1}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{1,1}$]	329
First	[**$J_{1,1}/M_{2,1}$**, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, $J_{5,1}/M_{3,1}$, $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{1,1}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{3,1}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{3,1}$]	1052
Second	[$J_{1,1}/M_{1,1}$, $J_{2,1}/M_{2,1}$, $J_{3,1}/M_{3,1}$, $J_{4,1}/M_{1,1}$, **$J_{5,1}/M_{2,1}$** , $J_{6,1}/M_{3,1}$, $J_{7,1}/M_{1,1}$, $J_{8,1}/M_{1,1}$, $J_{9,1}/M_{2,1}$, $J_{10,1}/M_{3,1}$, $J_{11,1}/M_{3,1}$]	579

6 THE OPTIMALITY OF TABU-SEARCH BASED HEURISTIC ALGORITHM

Demonstrating the efficacy of the proposed heuristic algorithm is an essential component of this research. This can be attained by assessing the quality of the final solution obtained by the algorithm and the total computation time it takes. The final solution evaluated by the heuristic algorithm can easily be assessed if the optimal solution is known. If the optimal solution is unknown, we need to compare the solution obtained by the heuristic algorithm to a suitable lower bound for the problem that is being investigated. The mathematical model developed in Chapter 4 can be used to quantify the effectiveness of the search algorithm by optimally solving small problem instances. As mentioned earlier, the mathematical model uses the branch-and-bound enumeration technique to obtain the optimal solution.

The example problem used in Chapter 5 is used again to show how a model can be formulated for a given problem instance. Recall there are two sets of binary variables - x_{ijg} and y_{ikjg}. The first variable x_{ijg} receives a value of 1 if job j is assigned to machine i of stage g or 0 otherwise. Generally, if each assignment of a job on a machine is considered, then there will be a total of $\sum_{g=1}^{G}(n \times m_g)$ variables, where n is the total number of jobs, G is the total number of stages and m_g is the number of machines in a given stage. Similarly, there will be a total of $\sum_{g=1}^{G}(n \times m_g)$ variables for c_{ijg} and $(n \times m_G)$ variables for t_{ijG}. Note that c_{ijg} and t_{ijG} are two sets of real variables. Some jobs cannot be processed on certain machines due to machine capability of capability.. Thus one can exclude the variables that correspond to those assignments. In a real problem, therefore, the total number of variables for x_{ijg} and c_{ijg} separately will be less than $\sum_{g=1}^{G}(n \times m_g)$ and the total number of variables for t_{ijG} will be less than $(n \times m_G)$. The second binary variable y_{ikjg} receives a value of 1 if job k precedes job j on machine i of stage g or 0 otherwise. There are a total of $\sum_{g=1}^{G}((n-1)/2 \times n \times m_g)$ variables for y_{ikjg} if all machines are assumed to be capable of processing all jobs.

A general model formulation for the example problem could have been developed where all eleven jobs could be processed on each machine in all the stages. This would have resulted in (11×19 + 5×11×19) = 1254 binary variables. This type of model formation is highly undesirable since mathematical models are computationally difficult to solve, particularly when they include a large number of binary variables. As we increase the number of variables, the computational time required to solve these problem instances becomes extremely large. Therefore, we formulate a more restricted (compact) model, i.e. a model that only incorporates feasible jobs-to-machine assignments, and allows jobs to skip stages. This type of model formulation results in a fewer number of variables and constraints. The compact model formulation provides a comprehensive insight into the research problem.

In order to identify the optimal solution for small problem instances, their corresponding formulated model was solved using the branch-and-bound enumeration method incorporated in CPLEX 9.0 (IBM, 2009) computer software. CPLEX (also referred to as ILOG CPLEX) was developed by Robert E. Bixby of *CPLEX* Optimization Inc. *CPLEX* Optimization was acquired by ILOG in 1997 and finally ILOG was acquired by IBM in 2009. The software was installed and run on an intel Core *i*3-370, 2.4GHz processor with 4 GB RAM. The large amount of time needed to identify the optimal solution is partly due to the large number of binary variables included in the model. CPLEX doesn't seem to be efficient enough even in solving small problem instances, although it uses the branch-and-bound technique, which is an implicit enumeration algorithm for solving combinatorial optimization problems.

In order to further examine the efficiency of CPLEX, ten problem instances were generated and run using CPLEX. These problem instances generated were large enough for CPLEX to solve. In other words, the total number of variables that are within the capacity of CPLEX 9.0 is 2100000000. The data generated for these problem instances used the same procedure as described in Section 7.1 of Chapter 7.

Table 6-1 Results of solving the problems implicitly using CPLEX 9.0

Problem instance	Number of Jobs	Number of Machines	Solution	Time (sec)
1	9	19	537	2080
2	10	19	1014	2643
3	10	19	472	4496
4	12	19	387	3822
5	12	19	963	5670
6	14	19	662	9076
7	16	19	Infeasible	28800
8	16	19	592	12178
9	18	19	Infeasible	28800
10	20	19	Infeasible	28800

6.1 Comparison Between the Optimal Solution and Solution Obtained by the Heuristic Algorithm

With the optimal solution obtained by CPLEX 9.0, the quality of the solution generated by the tabu-search based heuristic algorithms can easily be assessed. As described in Chapter 5, the search heuristics begin with an initial solution. Five different initial solutions were developed, namely: EDD (Earliest Due Date), LFJ/LFM (Least Flexible Job/Least Flexible Machine), LWT (Lowest Weighted Tardiness), DDW (Due Date Weight Ratio), HCR (Hybrid Critical Ratio). EDD, LFJ/LFM, LWT, DDW and HCR will be referred as IS1, IS2, IS3, IS4 and IS5 respectively. The initial solution generated by each of these methods is used as a starting point for the tabu-search based heuristic. Tabu search has few features that affect its performance as a heuristic algorithm. These features include short-term/long-term memory function and fixed/variable size of tabu list. There are two different approaches in the application of long-term memory function: the maximum frequency and the minimum frequency. The heuristic algorithms developed in this research encompass the combinations of these features, as shown in Table 6.2.

Table 6-2 Tabu search based heuristic algorithms used in this research

Types of Heuristic	Memory Function	Size of Tabu List
TS1	Short	Fixed
TS2	Long-Max	Fixed
TS3	Long-Min	Fixed
TS4	Short	Variable
TS5	Long-Max	Variable
TS6	Long-Min	Variable

Each initial solution method (IS) is used in combination with each type of tabu-search heuristics (TS). Thus there are a total of 30 heuristic combinations. Each combination is tested on 7 problem instances presented in Table 6.1. The remaining three problem instances are not used since CPLEX 9.0 couldn't identify the optimal solution for them, and thus there is no basis for comparison. The solutions obtained by the algorithm are then compared to the corresponding optimal solutions obtained by CPLEX 9.0. The percentage deviation of the algorithms from the optimal solutions is evaluated and reported in Table 6.3. Table 6.4 shows the computation time of each algorithm. The computation time presented in the table is the sum of time IS takes to generated the initial solution and the time TS takes to complete the search.

Table 6-3 Percentage deviation of the solutions obtained by the heuristics for small problems

Problem		**TS1**					**TS2**			
Instance	IS1	IS2	IS3	IS4	IS5	IS1	IS2	IS3	IS4	IS5
9 Job	3.1	0.6	0.0	3.7	6.2	3.1	3.1	0.0	3.7	2.6
10 Job	2.6	15.1	26.5	5.3	8.9	2.6	15.1	24.0	3.7	14.4

10 Job	0.0	14.7	11.1	0.0	12.4	0.0	0.5	13.4	0.0	7.8
12 Job	2.9	4.3	7.3	5.3	5.1	2.9	2.4	7.4	8.6	2.9
12 Job	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
14 Job	6.6	16.8	23.5	4.8	22.8	5.7	14.7	11.3	4.8	3.1
16 Job	0.7	5.5	7.2	6.6	0.0	3.1	5.5	7.2	0.6	0.0
Average	**2.27**	**8.14**	**10.80**	**3.67**	**7.91**	**2.49**	**5.90**	**9.04**	**3.06**	**4.40**
Problem		**TS3**					**TS4**			
Instance	IS1	IS2	IS3	IS4	IS5	IS1	IS2	IS3	IS4	IS5
9 Job	3.1	3.1	3.7	3.7	5.5	3.1	0.6	0.0	3.7	6.2
10 Job	2.6	18.2	6.3	3.0	7.3	2.6	15.1	26.5	5.3	8.9
10 Job	0.0	3.5	14.9	0.0	14.2	0.0	11.3	11.1	0.0	15.1
12 Job	3.7	1.5	2.7	7.5	8.0	2.9	4.3	9.3	5.3	4.7
12 Job	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
14 Job	7.2	15.2	10.1	4.8	17.1	6.0	13.2	23.5	4.4	17.8
16 Job	3.1	6.1	7.2	1.4	2.1	0.7	5.5	8.2	6.6	0.0
Average	**2.81**	**6.80**	**6.41**	**2.91**	**7.74**	**2.19**	**7.14**	**11.23**	**3.61**	**7.53**
Problem		**TS5**					**TS6**			
Instance	IS1	IS2	IS3	IS4	IS5	IS1	IS2	IS3	IS4	IS5
9 Job	3.1	2.1	0.0	3.7	2.6	3.1	3.1	3.7	3.7	5.5
10 Job	2.6	13.1	18.2	3.7	14.4	2.6	18.2	6.3	4.5	7.3
10 Job	0.0	1.5	9.1	0.0	7.8	0.0	3.5	14.9	0.0	14.2
12 Job	2.9	2.4	7.4	6.5	2.9	3.7	1.5	12.4	8.4	8.0
12 Job	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
14 Job	5.7	14.7	7.1	4.8	3.1	7.2	15.2	20.1	4.8	17.1
16 Job	3.1	6.9	7.2	0.6	0.0	3.1	6.1	7.2	4.1	2.1
Average	**2.49**	**5.81**	**7.00**	**2.76**	**4.40**	**2.81**	**6.80**	**9.23**	**3.64**	**7.74**

The average percentage deviation of all heuristic combination is 5.63% with 8 heuristic combinations below 3%. From the 30 heuristic combinations, IS1/TS4 appears to be the most effective heuristic combination in identifying the optimal solutions. The average percentage deviation for IS1/TS4 is 2.19%. The next best performer is IS1/TS1, which has an average percentage deviation of 2.27%. Both these heuristic combination use IS1

(Earliest Due Date) method as the initial solution generating mechanism, and basic search with fixed (TS1) and variable (TS4) tabu list size.

Table 6-4 Computation time of the heuristics for small problems (in seconds)

Problem		TS1					TS2			
Instance	IS1	IS2	IS3	IS4	IS5	IS1	IS2	IS3	IS4	IS5
9 Job	0.025	0.031	0.031	0.034	0.056	0.038	0.065	0.036	0.031	0.073
10 Job	0.041	0.082	0.065	0.075	0.071	0.033	0.054	0.064	0.082	0.054
10 Job	0.079	0.074	0.054	0.085	0.067	0.065	0.066	0.054	0.054	0.056
12 Job	0.038	0.068	0.043	0.067	0.07	0.055	0.032	0.068	0.068	0.056
12 Job	0.034	0.051	0.045	0.075	0.075	0.068	0.084	0.082	0.048	0.043
14 Job	0.046	0.049	0.061	0.068	0.087	0.057	0.049	0.067	0.096	0.058
16 Job	0.114	0.214	0.207	0.156	0.246	0.136	0.215	0.222	0.124	0.250
Average	0.054	0.081	0.072	0.080	0.096	0.065	0.081	0.085	0.072	0.084
Problem		TS3					TS4			
Instance	IS1	IS2	IS3	IS4	IS5	IS1	IS2	IS3	IS4	IS5
9 Job	0.032	0.085	0.049	0.057	0.054	0.030	0.056	0.046	0.056	0.046
10 Job	0.047	0.049	0.067	0.070	0.020	0.016	0.078	0.033	0.071	0.033
10 Job	0.088	0.067	0.024	0.066	0.071	0.055	0.036	0.065	0.067	0.065
12 Job	0.036	0.024	0.066	0.082	0.027	0.047	0.034	0.040	0.066	0.040
12 Job	0.07	0.064	0.046	0.088	0.056	0.03	0.079	0.08	0.067	.078.
14 Job	0.034	0.07	0.06	0.068	0.097	0.016	0.081	0.044	0.066	0.078
16 Job	0.217	0.159	0.221	0.174	0.250	0.188	0.245	0.203	0.185	0.218
Average	0.075	0.074	0.076	0.086	0.082	0.055	0.087	0.073	0.083	0.080
Problem		TS5					TS6			
Instance	IS1	IS2	IS3	IS4	IS5	IS1	IS2	IS3	IS4	IS5
9 Job	0.053	0.046	0.040	0.062	0.046	0.075	0.056	0.046	0.037	0.061
10 Job	0.088	0.046	0.046	0.062	0.046	0.064	0.071	0.046	0.03	0.096
10 Job	0.054	0.089	0.046	0.062	0.089	0.070	0.067	0.089	0.016	0.048
12 Job	0.036	0.04	0.064	0.020	0.04	0.020	0.066	0.040	0.055	0.046
12 Job	0.016	0.048	0.067	0.078	0.065	0.049	0.072	0.098	0.038	0.059
14 Job	0.055	0.065	0.058	0.061	0.054	0.055	0.090	0.044	0.061	0.041
16 Job	0.167	0.265	0.235	0.231	0.300	0.184	0.245	0.222	0.199	0.286
Average	0.067	0.086	0.079	0.082	0.091	0.074	0.095	0.084	0.062	0.091

6.2 The Effectiveness of Tabu-Search Based Heuristics for Medium and Large Problems

The computational complexity of the research problem being investigated has already been highlighted in chapter 4. The branch-and-bound enumeration technique can solve small problem instances but it is inefficient in finding the optimal solution for medium and large problem instances. As the problem structure grows larger, the branch-and-bound technique often fails to identify the optimal solution. Therefore to quantify the effectiveness of the solution algorithm for medium and large problem instances, we need to identify a suitable lower bound. However, the problem structure doesn't seem to lend itself to conveniently identify a lower bound. An alternative approach to evaluate the effectiveness of the heuristics for medium and large problem instances is by testing the heuristics on *carefully constructed* problem instances with a known *optimal* total weighted tardiness (TWT) of zero. The effectiveness of the heuristics can then be evaluated by measuring the deviation of their total weighted tardiness from the optimum solution (which is zero). A problem instance with an optimal TWT of zero can be generated using the following procedure:

1. Generate a problem instance using steps 1 to 9 of the procedure outlined in section 7.1 of chapter 7.
2. Randomly assign each job to a machine and record the completion time of jobs at the end of the last stage.
3. Set the due dates of all the jobs equal to their completion time in the last stage.

For medium and large problem structures, 4 problem instances are generated using the above procedure. There are a total of 30 heuristic combinations (5 levels of IS and 6 levels of TS) that need to be tested. The actual number of medium and large problems tested is 120 (30*4). The combination of IS1-IS5 and TS1-TS6 are applied to each problem instance. The values obtained from the heuristic combinations are compared to the TWT of the optimal schedule which is zero. Notice that the evaluation of the percentage deviation is not possible since it would lead to a division by zero (because the optimal solution is zero). In order to overcome this problem, the point of reference, which is the TWT of the optimal schedule, must be a positive value. In order to obtain a positive point of reference, the reference point needs to be shifted to a positive value. This can be attained by delaying the completion time of jobs in the last stage by one unit of time. As a result, the TWT of the optimal solution (which was zero before) will hold on to a positive value. This would result in a TWT that is equal to the sum of weights of all jobs. This positive TWT (obtained by offsetting the due dates of all jobs by 1) is used as the new reference point for evaluating the percentage deviation for the heuristic algorithm. Thus the percentage deviation is evaluated as:

$$\text{Percentage Deviation} = \begin{cases} \dfrac{\text{TWT} - \text{reference point}}{\text{reference point}} * 100\% & \text{if TWT} > \text{reference point} \\ 0 & \text{if TWT} \leq \text{reference point} \end{cases}$$

The results of applying the heuristics to the four medium problem structures are presented in Table 6.5. The first seven columns show the TWT obtained by each heuristic combination. The last seven columns show the percentage deviation of the TWT obtained by each heuristic combination. More than half of the heuristic combinations for the 25 job problem are able to identify the true optimal TWT of zero. Most of the heuristic combinations (across all problems) are able to identify a solution less than the reference point.

Table 6-5 Results of applying the heuristics to medium problem structures with zero values of TWT

25 Jobs, 17 Stages, 19 Machines (Reference Point = 48)													
TWT	TS1	TS2	TS3	TS4	TS5	TS6	% DEV	TS1	TS2	TS3	TS4	TS5	TS6
IS1	0	0	0	0	0	0	IS1	0.00	0.00	0.00	0.00	0.00	0.00
IS2	0	0	8	0	9	8	IS2	0.00	0.00	0.00	0.00	0.00	0.00
IS3	0	0	0	0	0	0	IS3	0.00	0.00	0.00	0.00	0.00	0.00
IS4	0	0	0	0	0	11	IS4	0.00	0.00	0.00	0.00	0.00	0.00
IS5	0	7	7	0	7	7	IS5	0.00	0.00	0.00	0.00	0.00	0.00
30 Jobs, 17 Stages, 19 Machines (Reference Point = 70)													

TWT	TS1	TS2	TS3	TS4	TS5	TS6	% DEV	TS1	TS2	TS3	TS4	TS5	TS6
IS1	49	16	16	16	16	16	IS1	0.00	0.00	0.00	0.00	0.00	0.00
IS2	47	47	47	47	47	47	IS2	0.00	0.00	0.00	0.00	0.00	0.00
IS3	57	56	56	57	56	56	IS3	0.00	0.00	0.00	0.00	0.00	0.00
IS4	14	11	11	14	11	11	IS4	0.00	0.00	0.00	0.00	0.00	0.00
IS5	36	21	18	36	21	18	IS5	0.00	0.00	0.00	0.00	0.00	0.00
35 Jobs, 17 Stages, 19 Machines (Reference Point = 72)													
TWT	TS1	TS2	TS3	TS4	TS5	TS6	% DEV	TS1	TS2	TS3	TS4	TS5	TS6
IS1	12	12	27	12	12	27	IS1	0.00	0.00	0.00	0.00	0.00	0.00
IS2	31	31	31	31	31	31	IS2	0.00	0.00	0.00	0.00	0.00	0.00
IS3	48	37	90	48	37	90	IS3	0.00	0.00	25.00	0.00	0.00	25.00
IS4	12	12	27	12	12	27	IS4	0.00	0.00	0.00	0.00	0.00	0.00
IS5	77	48	73	77	48	73	IS5	6.94	0.00	1.39	6.94	0.00	1.39
40 Jobs, 17 Stages, 19 Machines (Reference Point = 86)													
TWT	TS1	TS2	TS3	TS4	TS5	TS6	% DEV	TS1	TS2	TS3	TS4	TS5	TS6
IS1	108	108	119	108	108	119	IS1	25.58	25.58	38.37	25.58	25.58	38.37
IS2	117	110	110	117	110	110	IS2	36.05	27.91	27.91	36.05	27.91	27.91
IS3	128	128	128	128	128	128	IS3	48.84	48.84	48.84	48.84	48.84	48.84
IS4	108	108	119	108	108	119	IS4	25.58	25.58	38.37	25.58	25.58	38.37
IS5	123	123	112	123	123	112	IS5	43.02	43.02	30.23	43.02	43.02	30.23

To view the performance of each heuristic combination, the average percentage deviation over the four problem instances is evaluated and presented in Table 6.6. Eight heuristic combinations, i.e. IS1/TS1, IS2/TS2, IS1/TS4, IS2/TS5 and IS4/TS1, IS4/TS2, IS4/TS4, IS4/TS5 have the same minimum average percentage deviation of 6.40%. The percentage deviation averaged over the four problem instances and the 30 heuristic combinations is 9.45%. Based on these results, one may surmise that the heuristics are sufficiently effective in identifying very good near optimal solutions, if not the optimal solutions, for the medium problem structure.

Table 6-6 Average percentage deviation of the solutions obtained by the heuristics for medium problem structure

Initial	**Tabu-Search based heuristics**					
Solution	TS1	TS2	TS3	TS4	TS5	TS6
IS1	6.40	6.40	9.59	6.40	6.40	9.59
IS2	9.01	6.98	6.98	9.01	6.98	6.98
IS3	12.21	12.21	18.46	12.21	12.21	18.46
IS4	6.40	6.40	9.59	6.40	6.40	9.59
IS5	12.49	10.76	7.91	12.49	10.76	7.91

A similar effort is made to assess the effectiveness of the heuristics in identifying optimal solutions for large problem structure. Four problem instances that range from 40 to 60 jobs were generated using the random generation mechanism. As mentioned before, the problems falling between 40 to 60 jobs are considered large. Cognizant of the fact that the computational effort to solve the large problems can take anywhere between 45 minutes to 1 hour and 30 minutes, testing on many problem instances can take up a large computational effort. Another important thing to consider is that there are 30 heuristic combinations for each problem instance. For each problem instance, an optimal schedule with a TWT of 0 is obtained by setting the due dates equal to the completion times of the jobs. All 30 heuristic combinations are applied to each problem instance and the TWT of the final solutions is obtained accordingly. The percentage deviation of the final solutions is evaluated the same way as in the medium problem structure.

The TWT and percentage deviation obtained by each heuristic combinations are reported in Table 6.7. For 45 jobs problem and the 60 jobs problem, all 30 heuristic combinations identified a TWT less than the reference point. For 50 jobs problem, the heuristic combinations involving IS2, IS3 and IS5 are able to identify a final solution less than the reference point of 107 where as for the 55 jobs problem, the heuristic combinations involving IS1 and IS4 are able to identify a final solution less than the reference point of 112.

Table 6-7 Results of applying the heuristics to large problem structures with zero values of TWT

45 Jobs, 17 stages, 19 machines (Reference Point = 92)													
TWT	TS1	TS2	TS3	TS4	TS5	TS6	% DEV	TS1	TS2	TS3	TS4	TS5	TS6
IS1	53	53	53	53	53	53	IS1	0.0	0.0	0.0	0.0	0.0	0.0
IS2	41	41	53	41	41	53	IS2	0.0	0.0	0.0	0.0	0.0	0.0
IS3	48	41	53	48	51	53	IS3	0.0	0.0	0.0	0.0	0.0	0.0
IS4	70	70	68	70	70	68	IS4	0.0	0.0	0.0	0.0	0.0	0.0
IS5	68	68	68	68	68	68	IS5	0.0	0.0	0.0	0.0	0.0	0.0
50 Jobs, 17 stages, 19 machines (Reference Point = 107)													
TWT	TS1	TS2	TS3	TS4	TS5	TS6	% DEV	TS1	TS2	TS3	TS4	TS5	TS6
IS1	144	129	150	144	129	150	IS1	34.6	20.6	40.2	34.6	20.6	40.2
IS2	85	80	77	85	80	77	IS2	0.0	0.0	0.0	0.0	0.0	0.0
IS3	90	63	93	90	63	93	IS3	0.0	0.0	0.0	0.0	0.0	0.0
IS4	150	147	153	150	147	153	IS4	40.2	37.4	43.0	40.2	37.4	43.0
IS5	87	87	87	87	87	87	IS5	0.0	0.0	0.0	0.0	0.0	0.0
55 Jobs, 17 stages, 19 machines (Reference Point = 112)													
TWT	TS1	TS2	TS3	TS4	TS5	TS6	% DEV	TS1	TS2	TS3	TS4	TS5	TS6
IS1	110	93	80	110	93	80	IS1	0.0	0.0	0.0	0.0	0.0	0.0
IS2	163	128	160	163	128	163	IS2	45.5	14.3	42.9	45.5	14.3	45.5
IS3	186	186	189	186	186	189	IS3	66.1	66.1	68.8	66.1	66.1	68.8
IS4	98	98	98	98	98	98	IS4	0.0	0.0	0.0	0.0	0.0	0.0
IS5	158	132	159	158	132	159	IS5	41.1	17.9	42.0	41.1	17.9	42.0
60 Jobs, 17 stages, 19 machines (Reference Point = 113)													
TWT	TS1	TS2	TS3	TS4	TS5	TS6	% DEV	TS1	TS2	TS3	TS4	TS5	TS6
IS1	57	28	23	57	28	23	IS1	0.0	0.0	0.0	0.0	0.0	0.0
IS2	31	31	31	31	31	31	IS2	0.0	0.0	0.0	0.0	0.0	0.0
IS3	68	68	77	68	68	77	IS3	0.0	0.0	0.0	0.0	0.0	0.0
IS4	57	28	57	57	28	57	IS4	0.0	0.0	0.0	0.0	0.0	0.0
IS5	76	76	76	76	76	76	IS5	0.0	0.0	0.0	0.0	0.0	0.0

The average percentage deviation of the solutions obtained by each heuristic combination is evaluated over all four-problem instances and summarized in Table 6.8. It is apparent from the table that the heuristic combination of IS2/TS2 and IS2/TS5 exhibit only 3.57 percentage deviations. Other heuristic combinations IS5/TS2 and IS5/TS5 have a percentage deviation less than 5%. IS1/TS2 and IS1/TS5 have an average percentage deviation of only 5.14%. The percentage deviation averaged over the four problems and the 30 heuristic combinations is 10.36%. Based on these results, one can conclude that the heuristics are very effective in identifying a very good near optimal solution even for the large problem instances.

Table 6-8 Average percentage deviation of the solutions obtained by the heuristics for large problem structure

Initial	**Tabu-Search based heuristics**					
Solution	TS1	TS2	TS3	TS4	TS5	TS6
IS1	8.64	5.14	10.05	8.64	5.14	10.05
IS2	11.38	3.57	10.71	11.38	3.57	11.38
IS3	16.52	16.52	17.19	16.52	16.52	17.19
IS4	10.05	9.35	10.75	10.05	9.35	10.75
IS5	10.27	4.46	10.49	10.27	4.46	10.49

7 RESULTS AND DISCUSSION

Recall from Chapter 6, the tabu-search based heuristic algorithms proved to be highly efficient in comparison to the implicit enumeration technique (namely branch-and-bound) in solving small problem structures. While the heuristic algorithms take less than 2 seconds to solve the problem, the branch-and-bound technique embedded in CPLEX can take as long as 50 minutes just to solve a small problem structure. Chapter 6 also illustrated the efficiency of the tabu search mechanism for problems that cannot be solved using the branch-and-bound enumeration technique. Based on those results, the tabu-search based heuristic algorithms can be conjectured to provide very good/near optimal solution, if not optimal, to problem structures with no known optimal solutions. The research question is now focused on evaluating the comparative performance of the tabu-search based heuristics, aided by initial solution generation methods. Precisely, the intent of this research is to evaluate the performance of each algorithm as the size of the problem structure grows from small to medium and then large.

The size of a problem structure is determined by the number of jobs, *n*. The size of the problem structures covered in this research is defined as follows:

Small size: up to 20 jobs

Medium size: 21-40 jobs

Large size: 41 or more jobs

These sizes are selected based upon the direct feedback from the company for which the research was carried out. These sizes are selected to cover a wide variety of scheduling problems encountered in industry practice. Wide variety of problems allow us to evaluate whether the computation time required to solve them using the algorithm lies within reasonable expectations. Most of the small problem structures can be solved in less than a second. The medium problem structures require less than 2 seconds to be solved. Solving a large problem structure may require as much as 15 seconds. The increase in the computation time is due to the increase in complexity of the problem, presented in the form of an enlarged search space. The increase in search space causes the algorithm to consider more neighborhood solutions before selecting the best solution and then applying the move that results in the best solution. The increase in search space also delays the termination of the search as more moves are required before the stopping criterion is activated.

Note that to comply with the industry requirements, the algorithms were developed in C# .NET. The data was read using Microsoft Excel spreadsheet and the final results were also stored and displayed using Microsoft Excel spreadsheet. This was also done to comply with the industry requirements. Much of the work in the industry is done using Microsoft Excel and the easy user interface provided by Microsoft Excel has made it really popular over the years. The employees of the company for which this research was carried out were also proficient at Microsoft excel. Hence they wanted the algorithm to interact with Microsoft Excel (to input data and to display results). However, reading data into Microsoft Excel from C# is much more time consuming than reading data from other sources (say an SQL database or a text file). Solving a large problem may take as long as two hour.

For demonstrating the efficiency of the algorithm, only the computation time while the algorithm is running is taken into consideration. In other words, the time taken to read the data from the excel file is disregarded because had this algorithm been developed using text files, the time consumed to read the file would have been significantly lower (only a few seconds). C#.NET offers a built-in function that can easily verify the time required to run a routine. Therefore, the time required to run each TS algorithm can be easily obtained.

Once the sizes of the problem structures are established, an experiment can be conducted to address the following research issues:

1. To analyze the performance of the five initial solution generation methods on each size of the problem structure.
2. To analyze the performance of the six tabu-search based heuristics on each size of the problem structure.

3. To examine if the performance of the six tabu-search based heuristics is affected by the initial solution generation methods used.

7.1 Data Generation

With the exception of the estimation of C_{max}, which is the maximum completion time (makespan) of all jobs released, we take advantage of the data generation methodology proposed by Logendran and Subur (2000). As mentioned earlier, the structure of a problem is defined by the total number of jobs. The data used in this research, namely the run time of jobs, sequence-dependent setup time for jobs, job release time, job weight, job due date, and machine availability are generated using a random number generation (?) procedure. The notation for total number of projects is *p* and total number of jobs is *n*. The methodology for generating each problem instance can be documented as follows:

(1) The total number of projects for a given problem is generated from uniformly distributed random numbers over the interval [1, 10]. These random numbers must be integers.

(2) Once the number of projects for a given problem instance is determined, the number of jobs within each project is generated from uniformly distributed random numbers over the interval [1, 15]. These random numbers must be integers. As the total number of jobs considered in this research has the most influence in the scheduling algorithm, the classification that we use is based upon the total number of jobs as follows:

1-20 jobs – small size problem

21-40 - medium size problem

41 and higher – large size problem

(3) The machines can be categorized as least, medium or most capable. For the first stage (which has three machines), three random numbers are generated from a uniform distribution in [1, 10] and the coefficients of machines capability (α_i) are calculated as follows: the first (α_i) is assigned to the first machine. The second (α_i) is assigned to the second machine and so on. The machine that receives the smallest (α_i) represents

the most capable machine and the machine with the largest (α_i) represents the least capable machine. For subsequent stages with one machine each, a uniformly distributed random number is generated between [0, 1]. If the random number has a value less than or equal to 1/3, then the machine is considered to be most capable. If the random number is between 1/3 and 2/3 then the machine is considered to be of medium capability and lastly if the random number is greater than 2/3, the machine is considered to be least capable.

(4) If a job is capable of being processed on a machine, then the run time of the job is a randomly generated number between [α_i + 21] and [α_i + 40]. This reasoning is justifiable since a highly capable machine (that has a lower α_i) will have a lower run time for a job than a machine that has medium or lower capability (meaning that the machine has a higher α_i)

(5) The sequence-dependent setup times are generated from a uniform distribution over [1, 40]. Using a wide range such as this ensures that a large number of setup times generated for the same job on a machine are truly different.

(6) The release times of jobs are generated from a Poisson process with a mean arrival rate of 5 per hour, assuming that the setup and run times are all expressed in minutes. A Poisson process can be used to model the arrival of jobs independently of each other. A Poisson process was used to generate job release time by Schutten and Leussink (1996). These random numbers must take integer values.

(7) Machine availability times are also generated from a Poisson process with a mean arrival rate of 5 per hour. Suresh and Chaudhuri (1996) used Poisson process to model the occurrence of machine non-availability. As was the case with release times of jobs, the random numbers generated for machine availability must also take integer values. Once we have availability times for all machines, we delay the machine availability time in stage 2 by cumulative machine availability time in stage 1. The rationale behind doing so originates from the fact that at the start of any given planning horizon, the machines in the first stage have reference jobs loaded on them. The machines should process these reference jobs before processing the jobs from the current planning horizon. The ideal thing would be to evaluate the run time and set up time of the reference jobs and delay the machine availability by that much time. But

our motivation is to solve the problem instance that belongs to the current planning horizon rather than grappling with a problem from previous planning horizon (which in this case is that represented by the reference job). Therefore without making the problem overly complicated, we delay the machine availability time in stage 2 by cumulative machine availability time in stage 1. The machine availability for following stages (stage 3 to stage 17) is also evaluated by sequentially adding the machine availability times.

(8) The generation of meaningful due dates for the random test problem is by far the most challenging than the other algorithmic parameters. The due dates play a vital role in the evaluation of the total weighted tardiness of the jobs. In the absence of meaningful due dates, the initial solution and the search algorithm will not be effective in identifying an optimal or near optimal solution. In the past, researchers have used tardiness factor (τ), range factor (R), and C_{max} (the maximum completion time (makespan) of all jobs released) to generate meaningful due dates. τ is defined as $\tau = 1-\bar{d} / C_{max}$, where $\bar{d}$ is the average due date and C_{max} is the estimated makespan. A large value of τ indicates tight due dates and a small τ signifies loose due dates. *R* provides the measure of variability and is defined as $R = (d_{max} – d_{min}) / C_{max}$ where d_{max} is the largest due date and d_{min} is the smallest due date. A large value of *R* ensures that the randomly generated due dates are distributed over a wide interval whereas a small value of *R* guarantees due dates within a restricted range. The due dates are generated from a composite uniform distribution based on R and τ (refer to Table 7.1 for due date classification). With probability τ, the due date is uniformly distributed over the interval $[\bar{d} - R\bar{d}, \bar{d}]$ and with probability $(1-\tau)$, the due date is uniformly distributed over the interval $[\bar{d}, \bar{d} + (C_{max} - \bar{d})R]$. The evaluation of C_{max} is described in Section 5.3.4.

Note that all the random numbers are generated from a uniform distribution except for job release time and machine availability times, which are generated from a Poisson distribution. A uniform distribution has been proven to be appropriate to model length or duration of a process whereas a Poisson distribution has been proven to be appropriate to model the occurrence of an event at a given point of time.

Table 7-1 Due date classification

τ	R	Degree of tightness	Width of range
0.2	0.2	Loose	Narrow
0.2	0.5	Loose	Medium
0.2	0.8	Loose	Wide
0.5	0.2	Medium	Narrow
0.5	0.5	Medium	Medium
0.5	0.8	Medium	Wide
0.8	0.2	Tight	Narrow
0.8	0.5	Tight	Medium
0.8	0.8	Tight	Wide

7.2 Design of Experiment

A multi-factor experimental design is employed to address research questions 1, 2 and 3. The total weighted tardiness and the total computation time of the algorithms are used as a basis for performance measurement. Two factors are used in the experiment, they are the initial solution generation methods (IS) and different types of tabu-search based heuristics (TS). There are a total of five different levels of IS and six different levels of TS.

Three different sizes of problem structures were defined in the beginning of this chapter. Within each size, there are different structures to consider based upon different number of jobs. Within a problem structure, one can generate different problem instances (test problems) using the procedure described in section 7.1. All the problems are randomly generated and no two problem instances are exactly the same. Thus an experiment involving various problem instances and various problem structures will have fairly large variability in results. This variation can be reduced by treating each problem instance as a block. Blocking the problem instance is necessary to eliminate the influence of the differences between the problem instances (caused by random generation of problems). Thus, the differences in the performances of the algorithms, if identified, can be wholly attributed to the effect of the algorithms and not to the difference between problem instances.

All 30 (5 levels of IS * 6 levels of TS) combinations of factors are tested in each block. At this point, the experimental design looks like a randomized complete block design. Randomized complete block design is one of the most widely used experimental designs. Blocking can be used to systematically eliminate the effect of nuisance factor on the statistical comparisons among treatments. Blocking is an extremely important design technique, used extensively in industrial experiments.

A completely randomized block design may be defined as a design in which treatments are assigned to the experimental units completely at random. The design is completely flexible, i.e., any number of treatments and any number of levels (?) per treatment may be used. For this research, we are interested in finding out the effect of the two primary factors i.e., IS and TS. As mentioned earlier, IS has 5 different levels and TS has 6 different levels. Note that both these factors are of equal interest in this research because the IS finding mechanism helps in initiating the search whereas the TS mechanism helps in identifying the final solution. Moreover, previous research has shown that a quality initial solution may lead to better quality final solution (Logendran and Subur, 2004). Therefore, equal importance is given to both of these factors (IS and TS) while performing the statistical experimentation. A completely randomized block design was used where randomization was performed within each block. The 30 combination of factors (5 levels of IS and 6 levels of TS) were randomly assigned to the problem instances within a block. Blocking was used to account for the variability induced by the randomly generated problem instances.

A more complex design such as split-plot design was not chosen to conduct the experimentation because both the factors (IS and TS) are of equal interest to us. Moreover, we were able to completely randomize the order of the runs for all 30 heuristic combinations (5 levels of IS and 6 levels of TS). If we had to give more importance to a particular factor of interest (say TS) then we could have modeled the experiment as a split-plot design rather than a completely randomized block design. But as mentioned before, we are equally interested in the effects of both factors. Hence, a completely randomized block design was chosen over a split-plot design.

The experiment includes all three sizes of problem structures. For small size category, three different problem structures are used; they are 9 jobs and 19 machines, 12 jobs and 19 machines, and 17 jobs and 19 machines. For medium and large size category, the types of problem structures are reduced to two. Two problem structures are used for medium size category: 25 jobs and 19 machines and 35 jobs and 19 machines. Similarly, two problem structures are used for large size category: 45 jobs and 19 machines and 55 jobs and 19 machines. This reduction in size is due to the extensive computation time required to solve the medium and large problem, as explained in the beginning of the chapter.

Within each problem structure, 5 problem instances are generated. Each problem instance is characterized by the combination of the due date tightness factor (R) and the due date range factor (τ) used to generate the due dates of jobs in the problem. The combination of R and τ determines the characteristics of the due dates, as documented in Table 5.1. In order to cover different characteristics of due dates, 5 combinations of R and τ are selected from Table 5.1. Each combination is used in each problem instance (block) as:

Block 1: $\tau = 0.2$ and $R = 0.8$,

Block 2: $\tau = 0.5$ and $R = 0.5$,

Block 3: $\tau = 0.8$ and $R = 0.2$,

Block 4: $\tau = 0.2$ and $R = 0.2$,

Block 5: $\tau = 0.8$ and $R = 0.8$,

The five combinations are used consistently over each problem structure. The data generated for the experiment using the procedure described in section 7.1 is presented in Table D.1-D.3 in Appendix D for all problem structures. The experiment is performed on intel Core I3 2.1 GHz machine with 4 GB RAM.

7.3 Experimental Results and Analysis

The results of the experimentation are presented in Table E.1 – Table E.3 of Appendix E for small, medium and large problems, respectively. The total weighted tardiness presented in the tables refer to the final best solution obtained by a tabu-search heuristic (TS) using the initial solution (generated by an initial solution generation method (IS)). The computation time is the total time taken by an initial solution generation method and a tabu-search based heuristic to identify the final solution. The summary of the results collected for each problem structure is shown in Table 7.1. The analysis of results will focus on the total weighted tardiness first and then on the computation time. The correlation coefficient for TWT and computation time came out to be (-.18). This value suggests that there is no correlation between TWT and computation time and therefore, we can perform individual analysis on TWT and computation time.

As the summary of results only shows the average of the total weighted tardiness (TWT), one cannot conclusively say which level of factors yield the minimum TWT. A thorough statistical analysis on TWT will help us to determine if a particular level of a factor is better than the rest. A preliminary data exploration is essential to examine the distribution of TWT, in order to perform a statistical analysis on the TWT. Statistical analysis methods such as t-test are very powerful tools if the data is normally distributed. Graphical tools such as box plots can be very useful to detect any departure from the assumption of normality. The box plots of the TWT for all the levels of IS and all the levels of TS are shown in Figure F.1-F.3 of Apendix F for small medium and large problem structures, respectively. These box plots are generated by using popular statistical analysis package called *R* version 2.12.0. *R* is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and Mac Operating System.

The plots initially show that the data distribution is highly skewed and long tailed, which implied departure from normality and equal variance assumption. This is due to the large inconsistency between the values of the TWT obtained using different combinations of due date tightness factor (τ) and due date range factor (R). A problem with small (τ) and small (R) would tend to yield a relatively small TWT whereas a problem with large (τ) and small (R) tend to yield a relatively large value of TWT. Due to non-normality of the data distribution and unequal variance, parametric methods such as *F*-test and *t*-test are not appropriate for analyzing the experimental results.

Kruskal-Wallis test is a nonparametric alternative to the usual analysis of variance. Kruskal-Wallis can be applied to test the null hypothesis that involves *only* single factor. In this research, we have two factors (IS and TS) with blocking. Therefore Kruskal-Wallis test cannot be used as an alternative procedure to the *F*-test analysis of variance. Friedman test is another alternative for the *F*-test which can be applied for analyzing a single factor in a randomized block experiment. Again, we cannot apply the Friedman test as we have two factors together with blocking instead of 1 factor with blocking. Other non-parametric tests involve signed ANOVA but it is not widely accepted in the statistical community therefore it is also ruled out. Moreover, non-parametric tests fail to predict the interactions correctly (Conover, 1999). Therefore we need to keep the analysis in the realm of parametric tests.

Notice that problem instances generated using $\tau = 0.2$ and R = 0.2 (Block 4) yield loose due dates. All search algorithms (TS1-TS6) in conjunction with the initial solution generating mechanisms (IS1-IS5) are able to identify a TWT of zero and this is true for each problem structure (small, medium and large). A TWT value of zero makes the data highly skewed. If we remove Block 4 from the analysis, the box plots show that the data is approximately normal with approximately equal variance. A TWT value of zero implies that none of the jobs are tardy. In an industrial setting, there would hardly be any instance where none of the jobs are tardy. If none of the jobs are tardy, it means that the problem is not *carefully* constructed, meaning not rigorous enough to emulate industrial settings. Therefore, Block 4 is eliminated in performing further analysis.

After eliminating Block 4, the data becomes approximately normal (though a transformation is required). We can now apply a two-way ANOVA with blocking to each problem structure. Three different hypotheses need to be tested and they can be stated as follows:

Hypothesis 1

H_0: There is no difference in the TWT obtained for the problem instances using the five initial solution generation methods (IS).

H_1: At least one of the initial solution generation methods tends to yield a smaller TWT than the others.

Hypothesis 2

H_0: There is no difference in the TWT obtained for the problem instances using the six tabu search heuristics (TS).

H_1: At least one of the tabu search heuristics tends to yield a smaller TWT than the others.

Hypothesis 3

H_0: There is no interaction between IS and TS.

H_1: There is interaction between IS and TS.

The results of the two-way ANOVA with blocking are discussed individually on each problem structure. Among the various statistical methods available for comparing the effects of a factor at different levels, the least significance difference (LSD) based on *t*-statistic can be regarded as the least conservative, and Tukey-Kramer's adjusted *P*-value can be regarded as most conservative method. Since our goal is to truly identify a clear difference between the levels of a factor, we select Tukey-Kramer's adjusted *P*-value method for the purpose of analysis. Thus, while a particular factor may be deemed as significant, the detailed analysis for identifying which factor levels contribute to this significance may indeed point to none because of the stringent difference sought by Tukey-Kramer's adjusted *P*-value method. In the following sections, the analysis of results obtained for each size of problem is presented separately.

7.3.1 Small Problem Structures

A preliminary data exploration is carried out in order to perform a statistical analysis on TWT for small problem structures. The box plots of the TWT for all levels of IS and all levels of TS are shown in Figure F.1 of Appendix F for small problem structures. The plots show that the data is highly skewed which implies severe departure from normality and unequal variance. To stabilize the spread of the data variance, a natural-logarithm data transformation is applied. After the transformation, the TWT has a normal shape and the variance is equally spread as shown in Figure F.4 of Appendix F. Since the normality assumption for the parametric statistical method is met, an analysis of variance (ANOVA) or F-test can be applied to the log-transformed TWT (LOG_TWT).

The ANOVA result for the TWT on small problem structures has been presented in table 7.2. We assume a significance level (α) of 5%, as is commonly done in experimental design. The results clearly indicate that neither the initial solution finding mechanism (Pr > F = 0.7349 > 5%) nor the Tabu search algorithms (Pr > F = 0.4203 > 5%) have proven to be significant at 5% significance level. The results also suggest that the interaction between IS finding mechanism and TS algorithms is not significant at 5% significance level (Pr > F = 1.0000 > 5%). We can infer that no IS finding mechanism or TS algorithm has contributed to identifying a better quality solution (i.e. a lower TWT) in small size problems.

Table 7-2 Results for small size problems from ANOVA for TWT

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
IS	4	0.8730	0.2180	0.5011	0.7349
TS	5	2.1690	0.4340	0.9957	0.4203
Block	3	264.7100	88.2370	202.5638	<.0001*
IS:TS	20	0.7570	0.0380	0.0869	1.0000
Residuals	327	142.4410	0.4360		

Table 7-3 Results for small size problems from ANOVA for CT

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
IS	4	8.2330	2.0581	5.5174	<.0002 *
TS	5	0.1840	0.0368	0.0988	0.9923
Block	3	7.8420	2.6140	7.0075	<.0001 *
IS:TS	20	0.5170	0.0259	0.0694	1.0000
Residuals	327	121.9790	0.3730		

Table 7-4 Differences of least square means for CT of small size problems (IS)

	diff	lwr	upr	p adj
IS2-IS1	0.2718	0.0002	0.5433	0.0497
IS3-IS1	0.3441	0.0726	0.6157	0.0052
IS4-IS1	0.0494	-0.2222	0.3209	0.9875
IS5-IS1	0.3608	0.0893	0.6324	0.0028
IS3-IS2	0.0723	-0.1992	0.3439	0.9493
IS4-IS2	-0.2224	-0.4939	0.0491	0.1655
IS5-IS2	0.0890	-0.1825	0.3606	0.8971
IS4-IS3	-0.2947	-0.5663	-0.0232	0.0258
IS5-IS3	0.0167	-0.2549	0.2882	0.9998
IS5-IS4	0.3114	0.0399	0.5830	0.0154

Since no significant difference exists between the levels of IS or the levels of the TS algorithm based on TWT, we venture into finding if differences exists between the levels of IS or levels of TS based on CT. As was the case with TWT for small problem structures, initial data exploration for CT reveals that the data is highly skewed with unequal variance and spread. A log transformation is applied to normalize the data and the box plots have been presented in Figure G.4 of Appendix G. Since the transformed data has normal distribution and approximately equal variance, ANOVA can be applied. The ANOVA result for the CT on small problem structures has been presented in table 7.3. The IS finding mechanism (Pr > F = .0002 < 5%) reports to be significant at 5%

significance level. On the other hand, the TS algorithm (Pr > F = .9923 > 5%) does not report to be significant at 5% significance level. The results also suggest that the interaction between IS finding mechanism and TS algorithms is not significant at 5% significance level (Pr > F = 1.0000 > 5%).

Since the IS finding mechanism is significant at 5% significance level, this raises an interesting research question as to whether there is a unique IS finding mechanism that eventually leads to consuming lower computation time. An extended analysis is performed to address that question. Note that the processing speed of the computer comprising of 2.4 GHz i3 core processer, 4 GB RAM memory, and Windows 7 operating system enable us to evaluate the final answer within a fraction of a second for small problem instances. Though the analysis shows the IS finding mechanisms to be significant, it should be noted that solving the small problem instances does not even take one second (for all IS finding mechanisms and the TS search algorithms). Such is the efficacy of the entire search algorithm.

Table 7.4 summarizes the differences of least square means for CT of small problems based on IS finding mechanism. The comparison shows that IS1 (EDD) takes less computation time than IS2 (LFJ/LFM), IS3 (LWT) and IS5 (HCR). Table 7.4 also suggests that IS4 (DDW Ratio) takes less computation time than IS3 (LWT) and IS5 (HCR). Note that IS1 prioritizes jobs based on earliest due date whereas IS4 prioritizes jobs based on due dates as well as their respective weights. In other words, both IS1 and IS4 finding mechanisms prioritize jobs based on their due dates (with IS4 also incorporating the weights) hence it is not surprising that both take lower computation time simultaneously. However, there is no difference between IS1 (EDD) and IS4 (DDW Ratio) based on the least square means for CT as summarized in Table 7.4. Thus, if the decision is to be purely based on CT, IS1 (EDD) is the recommended choice since it can help save on the computation time over other IS finding mechanisms.

7.3.2 Medium Test Problems

Total Weighted Tardiness

A preliminary data exploration is carried out in order to perform a statistical analysis on TWT for medium problem structures. The box plots of the TWT for all levels of IS and all levels of TS are shown in Figure F.2 of Appendix F for medium problem structures. As was the case with small problem instances, the plots show that the data is highly skewed which implies severe departure from normality and unequal variance. To stabilize the spread of the data variance, a natural-logarithm data transformation is applied. After the transformation, the TWT has a normal shape and the variance is equally spread as shown in Figure F.5 of Appendix F. Since the normality assumption for the parametric statistical method is met, an analysis of variance (ANOVA) or F-test can be applied to the log-transformed TWT (LOG_TWT).

The ANOVA result for the TWT on medium problem structures has been presented in table 7.5. The results clearly suggests that neither the initial solution finding mechanism (Pr > F = 0.1101 > 5%) nor the Tabu search algorithms (Pr > F = 0.8431 > 5%) have proven to be significant at 5% significant level. The results also suggest that the interaction between IS finding mechanism and TS algorithms is not significant at 5% significance level (Pr > F = 1.0000 > 5%). We can infer that no IS finding mechanism or TS algorithm has contributed to identifying a better quality solution (i.e. a lower TWT) in medium?? size problems. In the absence of a distinct outperformer, TS1 with short-term memory and fixed TLS are recommended for medium problem instances.

Table 7-5 Results for medium size problems from ANOVA for TWT

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
IS	4	1.2900	0.3230	1.9093	0.1101
TS	5	0.3500	0.0690	0.4078	0.8431
Block	3	499.0800	166.3610	982.8069	< 0.0001
IS:TS	20	0.4800	0.0240	0.1432	1.0000

Residuals	207	35.0400	0.1690

Computation Time

The performance of each level of IS and TS is evaluated based on the computation time. Similar to TWT, an initial data exploration is performed on the computation time as the second response variable in the experiment. The box plots of computation time for medium problem structures are shown in Figure G.2 of Appendix G. The plots show that the distribution of computation time is highly skewed and the variance of each level of factor (i.e. IS or TS) is not equally spread. This is because some levels of IS or TS tend to take computation times that are much higher than the other levels. To stabilize the spread of data variance, a natural-logarithm data transformation is applied. The distribution of the transformed computation time has a normal shape and the variance is equally spread as shown in Figures G.5 of Appendix G. Since the normality assumption for parametric statistical methods is met, an analysis of variance (ANOVA) or F-test can be applied to the log-transformed computation time (LOG_CT).

The ANOVA result for the CT on medium problem structures has been presented in table 7.7. For medium size problems, the IS finding mechanism does not show any evidence of a difference at various levels (Pr > F = .9903 > 5%). However, the TS algorithm proves to be significant at 5% significance level (Pr > F = .0143 < 5%). The interaction between the IS finding mechanism and TS algorithm is also insignificant (Pr > F = 1.0000).

Further analysis focused only on the differences between the levels of TS algorithm produces no significant difference (Table 7.7) for any one TS algorithm to claim superiority of lower CT. As mentioned earlier, the least significance differences are evaluated based Tukey-Kramer’s adjusted P-value, which is the most conservative method. Thus, while a particular factor may be deemed as significant, the detailed analysis for identifying which levels contribute to this significance may indeed point to none because of the stringent difference sought by Tukey-Kramer’s adjusted P-value method. Though the plots clearly indicate that TS1 and TS4 probably require lower computation time than TS2, TS3, TS5 or TS6, the Tukey-Kramer's adjusted P-value method does not distinctively point out this significance at 5% significance level. In the absence of a statistically proven outperformer, TS1 with short-term memory and fixed TLS is recommended for large problems.

Table 7-6 Results for medium size problems from ANOVA for CT

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
IS	4	0.3960	0.0990	0.0729	0.9903
TS	5	19.8180	3.9636	2.9215	0.0143 *
Block	3	2.5790	0.8596	0.6336	0.5942
IS:TS	20	1.5040	0.0752	0.0554	1.0000
Residuals	207	280.8390	1.3567		

Table 7-7 Differences of least square means for CT for medium size problems

	diff	lwr	upr	p adj
TS2-TS1	-0.0669	-0.3315	0.1978	0.9784
TS3-TS1	-0.0137	-0.2783	0.2510	1.0000
TS4-TS1	0.0262	-0.2385	0.2908	0.9997
TS5-TS1	-0.0735	-0.3382	0.1911	0.9674
TS6-TS1	0.0116	-0.2531	0.2762	1.0000
TS3-TS2	0.0531	-0.2114	0.3178	0.9924
TS4-TS2	0.0930	-0.1716	0.3577	0.9138
TS5-TS2	-0.0067	-0.2713	0.2580	1.0000
TS6-TS2	0.0785	-0.1862	0.3431	0.9570
TS4-TS3	0.0398	-0.2248	0.3045	0.9980
TS5-TS3	-0.0598	-0.3245	0.2048	0.9869
TS6-TS3	0.0253	-0.2394	0.2899	0.9998
TS5-TS4	-0.0997	-0.3643	0.1649	0.8875
TS6-TS4	-0.0146	-0.2792	0.2501	1.0000
TS6-TS5	0.0851	-0.1795	0.3497	0.9397

7.3.3 Large Problem Structure

Total Weighted Tardiness

In order to perform a statistical analysis on TWT, a preliminary data exploration is necessary to examine the distribution of TWT (for large problems). The box plots of the TWT for all levels of IS and all levels of TS are shown in Figure F.3 of Appendix F for large problem structures. The plots show that the data is highly skewed, which implies severe departure from normality and unequal variance. To stabilize the spread of the data variance, a natural-logarithm data transformation is applied. After the transformation, the TWT has a normal shape and the variance is equally spread as shown in Figure F.6 of Appendix F. Since the normality assumption for the parametric statistical method is met, an analysis of variance (ANOVA) or F-test can be applied to the log-transformed TWT (LOG_TWT).

The ANOVA result for the TWT on large problem structures has been presented in table 7.8. The results clearly suggests that neither the initial solution finding mechanism (Pr > F = 0.8153> 5%) nor the Tabu search algorithms (Pr > F = 0.9036 > 5%) have proven to be significant at 5% significant level. The results also suggest that the interaction between IS finding mechanism and TS algorithms is not significant at 5% significance level (Pr > F = 1.0000 > 5%). We can infer that no IS finding mechanism or TS algorithm has contributed to identifying a better quality solution (i.e a lower TWT) in large size problems.

Table 7-8 Results for large size problems from ANOVA for TWT

	Df	**Sum Sq**	**Mean Sq**	**F value**	**Pr(>F)**
IS	4	1.0000	0.2490	0.3905	0.8153
TS	5	1.0000	0.2010	0.3150	0.9036
Block	3	485.0400	161.6810	253.4232	<.0001

IS:TS	20	0.8200	0.0410	0.0642	1.0000
Residuals	207	132.0600	0.6380		

Computation Time

Next, we determine if any IS finding mechanism or TS algorithm outperforms others based on the efficiency parameters. An initial data exploration is performed on the computation time for large problem structures. The box plots of computation time for different levels of IS mechanism and TS algorithm are shown in Figures G.3 of Appendix G for large problem structure. Recall, that natural logarithmic transformation was performed on CT for small and medium problem structures. But the plots for large problem structures show that the distribution of computation time (without the natural logarithmic transformation) is fairly normal and has reasonably equal spread for IS finding mechanism. Although the CT plot for TS algorithm is slightly skewed than the IS finding mechanism, for practical purposes, it can be assumed to abide by normality assumptions. The computation time required for small and medium problem structures was very small (in most cases it was less than a second), which often resulted skewed data with unequal variance. As the problem structure grows, the computation time required to solve the problem increases significantly, giving a better spread and fairly equal variance. Therefore the CT data for large problem structures does not require any transformation whereas the small and medium problem structures required a natural logarithmic transformation. The plots showing distribution of the computation time is shown in Figures G.3 of Appendix G. Since the normality assumption for parametric statistical methods is met, an analysis of variance (ANOVA) or F-test can be applied to the CT data.

Table 7-9 Results for large size problems from ANOVA for CT

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
IS	4	22.6000	5.6550	0.2487	0.9102
TS	5	659.0000	131.7980	5.7954	<.0001

Block	3	2.0000	0.6660	0.0293	0.9932
IS:TS	20	23.6000	1.1790	0.0518	1.0000
Residuals	207	4707.6000	22.7420		

The result for the CT on large problem structures has been presented in table 7.9. Clearly, the IS finding mechanism (Pr > F = .9102 > 5%) is not significant at 5% significance level. On the other hand, the TS algorithm (Pr > F = .0001< 5%) reports to be significant at 5% significance level. Therefore, further analysis is focused to find out if there is a significant difference between the levels of TS for large problems??. The box plots shown in Figure G.3 of Appendix G suggest that TS1 takes less computation time than TS2, TS3, TS5 and TS6 whereas TS4 takes less computation time than TS2, TS3, TS5 and TS6. Table 7.10 summarizes the differences of least square means for CT of large problems based on TS algorithms.

Table 7-10 Differences of least squares means for CT of large size problems

	diff	lwr	upr	p adj
TS2-TS1	3.7395	0.6720	6.8070	0.0073
TS3-TS1	3.4428	0.3753	6.5102	0.0179
TS4-TS1	0.3203	-2.7472	3.3877	0.9997
TS5-TS1	3.7983	0.7308	6.8657	0.0060
TS6-TS1	3.6675	0.6000	6.7350	0.0091
TS3-TS2	-0.2968	-3.6610	2.7707	0.9998
TS4-TS2	-3.4193	-6.4867	-0.3518	0.0192
TS5-TS2	0.0588	-3.0087	3.1262	1.0000
TS6-TS2	-0.0720	-3.1395	2.9955	1.0000
TS4-TS3	-3.1225	-6.1900	-0.0550	0.0434
TS5-TS3	0.3555	-2.7120	3.4230	0.9994
TS6-TS3	0.2248	-2.8427	3.2922	0.9999
TS5-TS4	3.4780	0.4105	6.5455	0.0161

TS6-TS4	3.3473	0.2798	6.4147	0.0235
TS6-TS5	-0.1308	-3.1982	2.9367	1.0000

The comparison shows that the pairs TS1-TS2, TS1-TS3, TS1-TS5 and TS1-TS6 are significantly different. The comparison also shows that the pairs TS4-TS2, TS4-TS3, TS4-TS5 and TS4-TS6 are significantly different. TS1 and TS4 are based on short term memory while others are based on LTM. In each of these pairs, the least squares mean estimate with short term memory produces a lower TWT than that with LTM. Thus, it pays to use short term memory search to solve large size problems. Since there is no significant difference between TS1 (short term memory with fixed TLS) and TS4 (short term memory with variable TLS), TS1 is recommended for efficiently solving the large size problems.

8 Conclusion and Suggestions for further research

A job scheduling problem in flexible flowshops with dynamic machine availability and dynamic job release time has been addressed in this research. In a flexible flowshop, one or more stages may have unrelated parallel machines. Unrelated parallel machines are machines that can perform the same function but have different capacity or capability. Since each machine has different capability, the run times of a job may differ from one machine to another. The machines considered in this research have dynamic availability time, which means that each machine may become available at a different time. The jobs are also assumed to be released dynamically. Each job in the scheduling problem considered in this research has a job release time, due date, and weight associated with it. A sequence-dependent setup time has also been considered in this research, which implies that a considerable amount of time can be spent to change over from one job to another. The release time can be viewed as a customer's order placement date, the due date can be considered as the shipment date and the weights can be considered as the priority of each job.

The possibility of machine skipping has also been incorporated in this research. A job may skip one or more stages depending upon customer's requirement or budgetary constraints. The objective of this research is to minimize the sum of weighted tardiness of all jobs released within the planning horizon. This research objective can be translated into on-time delivery or meeting customer's due dates. Such an objective is very important in industry practice because on-time delivery is the underlying factor for customer's satisfaction.

The research problem is formulated as a mixed (binary) integer-linear programming model with the objective function focused on minimizing the total weighted tardiness of all jobs released. The computational complexity of the research problem is shown to be strongly NP-hard. Because it is strongly NP-hard, an implicit enumeration technique can only be used to solve small problem instances in reasonable computation time. For medium and large problem instances, the branch and bound technique would not only be very time consuming, but in some cases may never find the optimal solution even after investing an exceedingly large computation time. Knowing the inefficiency of the implicit enumeration method, a higher-level search heuristic, based on the concept of tabu search, is developed and applied to solve the research problem.

Six different tabu-search based heuristics are developed by incorporating the different features of tabu search such as short and long term memory with fixed and variable tabu-list size. Five different methods are developed to generate the initial solution that can be used as starting points by tabu search. Two of the initial solutions are developed based on simple dispatching rules known as the Earliest Due Date (EDD) and Due Date Weight Ratio (DDW). The difference between the two methods is that the former is myopic to weights. A more complex initial solution presented in this research was the hybrid critical ratio (HCR), which is a modified form of DDW and takes into account the set-up times and runtimes of jobs besides due date and weight. LFJ/LFM and LWT are two other initial solutions presented, which are based on flexibility (of jobs and machines) and makespan of jobs, respectively.

In order to assess the quality of the final solutions obtained from tabu-search based heuristics, ten small problem instances were generated and solved using the branch-and bound technique embedded in CPLEX 9.0 and the tabu search based heuristics. Using the branch-and bound technique, 7 out of 10 problem instances were solved optimally. The optimal solutions are then compared with the solutions obtained from the tabu-search based heuristics. One of the heuristics (IS1/TS4) obtained solutions that have average percentage deviation of only 2.19%, thus demonstrating the capability of search heuristics to identify high quality solutions.

Since the optimal solutions for the medium and large problem structures are not attainable, the effectiveness of the tabu-search based heuristics is evaluated differently from small problem structure. Four problem instances of medium size and four problem instances of large size were constructed to have zero total weighted tardiness. Since the optimal solutions for these problem instances have zero total weighted tardiness, the point of reference for evaluating the deviation is shifted to a positive value. This reference point is obtained by delaying the completion times of all jobs in the optimal schedule by one unit of time. Thus, the percentage deviation of the solutions obtained by the heuristics was evaluated based on the reference point. The results show that the average percentage deviation evaluated over the heuristics is 9.45% for the medium problem structure and 10.36% over the large problem structure.

A more complete experiment with a broader scope was conducted to assess the performance of the heuristics as the size of the problem grows from small to medium and finally to large. A multi-factor experiment with randomized complete block design was conducted. The design of the experiment included two different factors. The five initial solution generation methods (IS1-IS5) were the levels of one factor and the six tabu-search heuristics (TS1-TS6) were the levels of the other factor. The total weighted tardiness and the computation time were the two performance measures used. The results of the experiment suggest that no IS finding mechanism or TS algorithm contributed to identifying a better quality solution (i.e a lower TWT) for all three problem instances (i.e. small, medium and large). In other words, no IS finding mechanism or TS algorithm could statistically outperform others. In absence of a distinct outperformer, TS1 with short-term memory and fixed TLS are recommended for all problem instances.

When comparing the efficiency of the search algorithms, the results of the experiment show that IS1, which refers to the EDD method, is recommended as the initial solution generation method for small problem sizes. The EDD method is capable of obtaining an initial solution that helps the tabu-search based heuristic to get to the final solution within a short time. TS1 is recommended as the tabu-search based heuristic for large problems, in order to save on time. TS1 is also recommended to solve small and medium problem structures in the absence of a statistically proven outperformer.

As mentioned before, this research focuses on minimizing the total weighted tardiness, in a flexible flowshop setting only. Further research may consider scheduling jobs in a job-shop environment since not all industrial settings tend to follow a flowline arrangement. Scheduling jobs in a job-shop environment will further add to the complexity of developing the search algorithm as well as the mathematical model. However, cognizant of the industrial significance of a job-shop setting, the proposed research may be relevant to various firms across the manufacturing industry.

Further research could also focus on comparing the performance of tabu search to other higher-level heuristics such as genetic algorithm and simulated annealing in solving the scheduling problems addressed in this research. The performance of these heuristics has been compared to tabu search in solving different types of scheduling problems (Park and Kim, 1997, Piersma and Van Dijk, 1996, Glass et al., 1994). These heuristics have shown different performances in different applications. More insights can be gained by applying simulated annealing or genetic algorithm to the research problem and comparing their results to the results obtained from this research.

Besides scheduling, rescheduling of jobs is of high relevance? in any manufacturing scenario. The ability to reschedule jobs gives the flexibility to introduce, modify or cancel jobs during a particular planning horizon. A job may have to be rescheduled depending upon the change in customer's demand, machine breakdowns or due to any unforeseen circumstances (labor or raw-material shortages).

BIBLIOGRAPHY

Azizoglu, M., and Kirca, O. "Scheduling Jobs On Unrelated Parallel Machines to Minimize Regular Total Cost Functions." IIE Transactions (1999): 153-159.

Barnes, J. W., M. Laguna, and F. Glover. "An Overview of Tabu Search Approaches to Production Scheduling Problems." Intelligent Scheduling Systems (1995): 101.

Bradley, J.V., Distribution-Free Statistical Tests (1968).

Brah, S. A., and J. L. Hunsucker. "Branch and Bound Algorithm for the Flow-Shop with Multiple Processors." European Journal of Operational Research 51.1 (1991): 88-99.

C#.NET Version 3.0 “Microsoft Visual Studio.NET” (2008).

Carlier, J. "Scheduling Jobs with Release Dates and Tails on Identical Machines to Minimize Makespan. " European Journal of Operational Research 29 (1987): 298-306.

Carroll, D.C. "Heuristic Sequencing of Jobs with Single and Multiple Components." PhD Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts (1965).

Centeno, G., and Armacost R. L. "Parallel Machine Scheduling with Release Time and Machine Eligibility Restrictions. " Computers and Industrial Engineering 33.1 (1997): 273-276.

Cheng, T. C. E., and C. C. S. Sin. "A State-of-the-Art Review of Parallel-Machine Scheduling Research." European Journal of Operational Research 47.3 (1990): 271-92.

Conover, W. J., Practical Nonparametric Statistics (1999) (New York: Wiley).

Conway, R. W., Maxwell, W. L., and Miller, L.W., Theory of Scheduling (1967) (Reading, MA: Addison-Wesley).

Davis, E., and Jaffe, J. M. "Algorithms for Scheduling Tasks on Unrelated Processor." Laboratory for Computer Science, Massachusetts Institute of Technology (1979).

Ding, F. Y., and D. Kittichartphayak. "Heuristics for Scheduling Flexible Flow Lines." Computers & Industrial Engineering 26.1 (1994): 27-34.

Glover, F. "Future Paths for Integer Programming and Links to Artificial-Intelligence." Computers & Operations Research 13.5 (1986): 533-49.

Guinet, A. "Scheduling Independent Jobs on Uniform Parallel Machines to Minimize Tardiness Criteria." Journal of Intelligent Manufacturing 6 (1995): 95-103.

Hariri, A. M. A., and Potts, C.N. "Heuristics for Scheduling Unrelated Parallel Machines." Journal of Computers and Operations Research 18.3 (1991): 323-331.

Ho, J. C., and Chang, Y. L. "Heuristics for Minimizing Mean Tardiness for m Parallel Machines." Naval Research Logistics 38 (1991): 367-381.

Hubscher, R., and Glover, F. "Applying Tabu Search with Influential Diversification to Multiprocessor Scheduling." Journal of Computers and Operations Research 21.8 (1994): 877-884.

Glass, C. A., Potts, C. N., and Shade, P. "Unrelated Parallel Machine Scheduling Using Local Search." Mathematical and Computer Modeling 20.2 (1994): 41-52.

Glover, F. "Future Paths for Integer Programming and Links to Artificial Intelligence." Computers and Operations Research 13.55 (1986): 533-549.

Glover, F. "Tabu Search-Part I." ORSA Journal on Computing 1.3 (1989): 190-206.

Glover, F. "Tabu Search: A Tutorial." Interfaces 20 (1990): 74-94.

Glover, F. "Tabu Search-Part II." ORSA Journal on Computing 2.1 (1990): 4-32.

IBM. (2009) *ILOG CPLEX Optimization Studio,* Version 12.2.

Jayamohan, M. S., and C. Rajendran. "A Comparative Analysis of Two Different Approaches to Scheduling in Flexible Flow Shops." Production Planning & Control 11.6 (2000): 572-80.

Jungwattanakit, J., et al. "A Comparison of Scheduling Algorithms for Flexible Flow Shop Problems with Unrelated Parallel Machines, Setup Times, and Dual Criteria." Computers & Operations Research 36.2 (2009): 358-78.

Comparing Scheduling Rules for Flexible Flow Lines. 2003. Elsevier Science Bv.

Karim, Y. "The Impact of Alternative Cell Locations and Alternative Routes of Material Handling Equipment in the Design of Cellular Manufacturing Systems." MS Thesis, Oregon State University, Corvallis, Oregon (1999).

Koulmas, C. "The Total Tardiness Problem: Review and Extensions." Operations Research 42.6 (1994): 1024-1041.

Kyparisis, G. J., and C. Koulamas. "A Note on Makespan Minimization in Two-Stage Flexible Flow Shops with Uniform Machines." European Journal of Operational Research 175.2 (2006): 1321-27.

Lam, K., and Xing, W. "New Trends in Parallel Machine Scheduling." International Journal of Operations and Production Management 17 (1997): 326-338.

Lee, Y. H., Bhaskaran, K., and Pinedo, M. "A Heuristic to Minimize the Total Weighted Tardiness with Sequence-Dependent Setups." IIE Transactions 29 (1997): 45-52.

Lenstra, J.K., Rinnooy Kan, A. H. G., and Brucker, P. "Complexity of Machine Scheduling Problems." Annals of Discrete Mathematics 1 (1977): 343-362.

Lenstra, J. K., Shmoys, D. B., and Tardos, E. "Approximation Algorithms for scheduling Unrelated Parallel Machines." Report OS-R8714 – Centre of Mathematics and Computer Science, Amsterdam (1987).

Logendran, R., and Sonthinen, A. "A Tabu Search-Based Approach for Scheduling Job-Shop type flexible manufacturing systems." Journal of the Operational Research Society 48 (1997): 264-277.

Lodree, E., W. S. Jang, and C. M. Klein. "A New Rule for Minimizing the Number of Tardy Jobs in Dynamic Flow Shops." European Journal of Operational Research 159.1 (2004): 258-63.

Logendran, R., P. deSzoeke, and F. Barnard. "Sequence-Dependent Group Scheduling Problems in Flexible Flow Shops." International Journal of Production Economics 102.1 (2006): 66-86.

Logendran, R., B. McDonell, and B. Smucker. "Scheduling Unrelated Parallel Machines with Sequence-Dependent Setups." Computers & Operations Research 34.11 (2007): 3420-38.

Logendran, R., and A. Sonthinen. "A Tabu Search-Based Approach for Scheduling Job-Shop Type Flexible Manufacturing Systems." Journal of the Operational Research Society 48.3 (1997): 264-77.

Logendran, R., and F. Subur. "Unrelated Parallel Machine Scheduling with Job Splitting." IIE Transactions 36.4 (2004): 359-72.

Montgomery, D. C. Design and Analysis of Experiments (New York: Wiley) (1991).

Muller, F. M., Franca P. M., and Guidini, C. M. "A Diversification Strategy for a Tabu Search Heuristic to Solve the Multiprocessor Scheduling Problem with Sequence Dependent Setup times." Proceeding of The 11th ISPE/IEE/IFAC International Conference on CAD/CAM Robotics and Factories of the Future (1995): 217-222.

Neave, H. R., and Worthington, P. L. Distribution-Free Tests (London: Unwin Hyman) (1988).

Ow, P. S., and Morton, T.E. "The Single Machine Early/Tardy Problem." Management Science 35.2 (1989): 177-191.

Park, M. W., and Kim, Y. D. "Search Heuristics for a Parallel Machine Scheduling Problem with Ready Times and Due Dates." Computers and Industrial Engineering 33.3 (1997): 793-796.

Piersma, N., and Van Dijk, W. "A Local Search Heuristic for Unrelated Heuristic for Unrelated Parallel machine Scheduling with Efficient Neighborhood search." Mathematical and Computer Modeling 24.9 (1996): 11-19.

A Diversification Strategy for a Tabu Search Heuristic to Solve the Multiprocessor Scheduling Problem with Sequence Dependent Setup Times. 1995.

Pfund, M., et al. "Scheduling Jobs on Parallel Machines with Setup Times and Ready Times." Computers & Industrial Engineering 54.4 (2008): 764-82.

R Statistical Software Version 2.11.0. “Support for Windows 64 bit systems” (2010).

Rachamadugu, R. V. and Morton, T. E. "Myopic Heuristics for the Single Machine Weighted Tardiness Problem." Master’s Thesis – Graduate School of Industrial Administration, Carnegie-Mellon University (1981): 81-82.

Salvador, M. S. "A Solution of Special Class of Flowshop Scheduling and Its Application." Berlin: Springer-Verlag, 1973.

Schutten, J. M. J., and Leussink, R. A. M. "Parallel Machine Scheduling with Release Dates, Due Dates, and Family Setup Times." International Journal of Production Economics 46/47 (1996) 119-125.

Sriskandarajah, C., and S. P. Sethi. "Scheduling Algorithms for Flexible Flowshops - Worst and Average Case Performance." European Journal of Operational Research 43.2 (1989): 143-60.

Suresh, V., and Chaudhuri, D. "Minimizing Maximum Tardiness for Unrelated Parallel Machines." International Journal of Production Economics 34 (1994): 223-229.

Suresh, V., and Chaudhari, D. "Bicriteria Scheduling Problem for Unrelated Parallel Machines." Computers and Industrial Engineering 30.1 (1996a): 77-82.

Suresh, V., and Chaudhari, D. "Bicriteria Scheduling Problem for Unrelated Parallel Machines." Computers and Industrial Engineering 30.1 (1996a): 77-82.

Suresh, V., and Chaudhari, D. "Scheduling of Unrelated Parallel Machines When Machine Availability is Specified." Production Planning and Control 7.4 (1996b): 393-400.

Tasgetiren, M. F., Q. K. Pan, and Y. C. Liang. "A Discrete Differential Evolution Algorithm for the Single Machine Total Weighted Tardiness Problem with Sequence Dependent Setup Times." Computers & Operations Research 36.6 (2009): 1900-15.

Vepsalainen, A. P. J., and Morton, T. E.

Wang, H. "Flexible Flow Shop Scheduling: Optimum, Heuristics and Artificial Intelligence Solutions." Expert Systems 22.2 (2005): 78-85.

APPENDICES

Appendix A. Setup Times for the example problem

Change over from $J_{j'p'}$ to $J_{1,1}$ on $M_{i,k}$																		
	$M_{1,1}$	$M_{2,1}$	$M_{3,1}$	$M_{1,2}$	$M_{1,3}$	$M_{1,4}$	$M_{1,5}$	$M_{1,6}$	$M_{1,7}$	$M_{1,8}$	$M_{1,9}$	$M_{1,10}$	$M_{1,11}$	$M_{1,12}$	$M_{1,13}$	$M_{1,14}$	$M_{1,15}$	$M_{1,16}$	$M_{1,17}$
$J_{1,1}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$J_{2,1}$	0	7	0	0	38	0	0	0	0	0	0	0	0	0	0	35	0	12	14
$J_{3,1}$	0	0	0	0	2	0	0	0	0	0	0	23	0	0	0	0	0	24	30
$J_{4,1}$	22	0	0	0	28	0	0	0	0	0	0	0	0	0	0	0	0	26	11
$J_{5,1}$	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	28	0	21	13
$J_{6.1}$	0	0	0	0	38	0	0	0	0	0	0	0	0	0	0	29	0	29	17
$J_{7,1}$	30	0	0	0	0	0	0	0	0	0	0	14	0	9	0	37	0	29	1
$J_{8,1}$	5	0	0	0	20	0	0	0	0	0	0	8	0	0	0	4	0	38	35
$J_{9,1}$	0	5	0	0	18	0	0	0	0	0	0	0	0	0	0	0	0	31	39
$J_{10,1}$	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	26	27
$J_{11,1}$	32	0	0	0	14	0	0	0	0	0	0	0	0	0	0	16	0	12	10

Change over from $J_{j'p'}$ to $J_{2,1}$ on $M_{i,k}$

	$M_{1,1}$	$M_{2,1}$	$M_{3,1}$	$M_{1,2}$	$M_{1,3}$	$M_{1,4}$	$M_{1,5}$	$M_{1,6}$	$M_{1,7}$	$M_{1,8}$	$M_{1,9}$	$M_{1,10}$	$M_{1,11}$	$M_{1,12}$	$M_{1,13}$	$M_{1,14}$	$M_{1,15}$	$M_{1,16}$	$M_{1,17}$
$J_{1,1}$	0	13	0	0	11	0	0	0	0	0	0	0	0	0	0	32	0	32	22
$J_{2,1}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$J_{3,1}$	0	0	0	0	14	0	0	0	0	0	34	0	0	0	0	0	39	15	4
$J_{4,1}$	0	0	0	0	34	0	0	0	0	0	0	0	0	0	0	0	14	23	12
$J_{5,1}$	0	39	0	0	0	2	0	0	0	0	0	0	0	0	0	13	17	7	24
$J_{6,1}$	0	0	0	0	20	26	0	0	0	0	0	0	0	0	0	17	26	16	32
$J_{7,1}$	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	12	0	18	29
$J_{8,1}$	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	35	16	19	4
$J_{9,1}$	0	23	0	0	6	0	0	0	0	0	17	0	0	0	0	0	20	10	23
$J_{10,1}$	0	0	0	0	24	0	0	0	0	0	0	0	0	0	0	0	21	21	8
$J_{11,1}$	0	0	0	0	24	0	0	0	0	0	16	0	0	0	0	3	3	31	7

Change over from $J_{j'p'}$ to $J_{3,1}$ on $M_{i,k}$																		
	$M_{1,1}$	$M_{2,1}$	$M_{3,1}$	$M_{1,2}$	$M_{1,3}$	$M_{1,4}$	$M_{1,5}$	$M_{1,6}$	$M_{1,7}$	$M_{1,8}$	$M_{1,9}$	$M_{1,10}$	$M_{1,11}$	$M_{1,12}$	$M_{1,13}$	$M_{1,14}$	$M_{1,15}$	$M_{1,16}$	$M_{1,17}$
$J_{1,1}$	0	0	0	0	6	0	0	0	0	0	0	15	0	0	0	0	0	17	15
$J_{2,1}$	0	0	0	0	27	0	0	0	0	0	22	0	0	0	0	0	7	37	17
$J_{3,1}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$J_{4,1}$	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0	0	18	21	3
$J_{5,1}$	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	13	23	28
$J_{6,1}$	0	0	20	0	2	0	0	0	0	0	0	0	0	0	0	0	10	33	2
$J_{7,1}$	0	0	6	0	0	0	0	0	0	0	0	4	0	0	0	0	0	9	26
$J_{8,1}$	0	0	0	0	20	0	0	0	0	0	0	13	0	0	0	0	29	26	24
$J_{9,1}$	0	0	14	0	14	0	0	0	0	0	17	0	0	0	0	0	22	5	12
$J_{10,1}$	0	0	1	0	4	0	0	0	0	0	0	0	0	0	0	0	11	38	19
$J_{11,1}$	0	0	6	0	31	0	0	0	0	0	22	0	0	0	0	0	11	37	24

Change over from $J_{j'p'}$ to $J_{4,1}$ on $M_{i,k}$

	$M_{1,1}$	$M_{2,1}$	$M_{3,1}$	$M_{1,2}$	$M_{1,3}$	$M_{1,4}$	$M_{1,5}$	$M_{1,6}$	$M_{1,7}$	$M_{1,8}$	$M_{1,9}$	$M_{1,10}$	$M_{1,11}$	$M_{1,12}$	$M_{1,13}$	$M_{1,14}$	$M_{1,15}$	$M_{1,16}$	$M_{1,17}$
$J_{1,1}$	31	0	0	0	22	0	0	0	0	0	0	0	0	0	0	0	0	20	17
$J_{2,1}$	0	0	0	0	14	0	0	0	0	0	0	0	0	0	0	0	26	17	12
$J_{3,1}$	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	4	24	27
$J_{4,1}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$J_{5,1}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	6	17	30
$J_{6,1}$	0	0	0	0	28	0	0	0	0	0	0	0	0	0	10	0	31	29	18
$J_{7,1}$	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	35	35
$J_{8.1}$	28	0	0	28	3	0	0	0	0	0	0	0	0	0	13	0	9	23	35
$J_{9,1}$	0	0	0	25	37	0	0	0	0	0	0	0	0	0	0	0	21	12	9
$J_{10,1}$	0	0	0	0	37	0	0	0	0	0	0	0	0	0	26	0	38	14	3
$J_{11,1}$	4	0	0	0	16	0	0	0	0	0	0	0	0	0	21	0	37	9	30

Change over from $J_{j'p'}$ to $J_{5,1}$ on $M_{i,k}$

	$M_{1,1}$	$M_{2,1}$	$M_{3,1}$	$M_{1,2}$	$M_{1,3}$	$M_{1,4}$	$M_{1,5}$	$M_{1,6}$	$M_{1,7}$	$M_{1,8}$	$M_{1,9}$	$M_{1,10}$	$M_{1,11}$	$M_{1,12}$	$M_{1,13}$	$M_{1,14}$	$M_{1,15}$	$M_{1,16}$	$M_{1,17}$
$J_{1,1}$	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	19	0	33	22
$J_{2,1}$	0	12	0	0	0	36	0	0	0	0	0	0	0	0	0	28	24	30	4
$J_{3,1}$	0	0	18	0	0	0	0	0	0	0	0	0	0	0	0	0	27	8	37
$J_{4,1}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	1	18	30
$J_{5,1}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$J_{6,1}$	0	0	25	0	0	37	0	0	0	0	0	0	0	0	21	2	6	28	17
$J_{7,1}$	0	0	15	0	0	29	0	0	0	0	0	0	0	0	0	21	0	11	12
$J_{8,1}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31	2	6	25	21
$J_{9,1}$	0	11	16	0	0	0	0	0	0	0	0	0	0	0	0	0	3	27	25
$J_{10,1}$	0	0	10	0	0	0	0	0	0	0	0	0	0	0	20	0	5	29	20
$J_{11,1}$	0	0	34	0	0	0	0	0	0	0	0	0	0	0	6	4	34	27	35

Change over from $J_{j'p'}$ to $J_{6,1}$ on $M_{i,k}$

	$M_{1,1}$	$M_{2,1}$	$M_{3,1}$	$M_{1,2}$	$M_{1,3}$	$M_{1,4}$	$M_{1,5}$	$M_{1,6}$	$M_{1,7}$	$M_{1,8}$	$M_{1,9}$	$M_{1,10}$	$M_{1,11}$	$M_{1,12}$	$M_{1,13}$	$M_{1,14}$	$M_{1,15}$	$M_{1,16}$	$M_{1,17}$
$J_{1,1}$	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	25	0	10	18
$J_{2.1}$	0	0	0	0	13	9	0	0	0	0	0	0	0	0	0	14	17	34	38
$J_{3,1}$	0	0	9	0	31	0	0	0	0	0	0	0	0	0	0	0	28	26	14
$J_{4.1}$	0	0	0	0	26	0	0	0	0	0	0	0	0	0	4	0	15	34	8
$J_{5,1}$	0	0	37	0	0	19	0	0	0	0	0	0	0	0	29	25	33	39	11
$J_{6.1}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$J_{7,1}$	0	0	18	0	0	38	0	0	0	0	0	0	0	0	0	8	0	35	38
$J_{8,1}$	0	0	0	0	8	0	0	0	0	0	0	0	0	0	4	26	1	30	9
$J_{9,1}$	0	0	19	0	10	0	0	0	0	0	0	0	0	0	0	0	19	5	15
$J_{10,1}$	0	0	25	0	4	0	0	0	0	0	0	0	0	0	18	0	28	22	16
$J_{11,1}$	0	0	15	0	22	0	0	0	0	0	0	0	0	0	10	3	3	14	30

										Change over from $J_{j'p'}$ to $J_{7,1}$ on $M_{i,k}$									
	$M_{1,1}$	$M_{2,1}$	$M_{3,1}$	$M_{1,2}$	$M_{1,3}$	$M_{1,4}$	$M_{1,5}$	$M_{1,6}$	$M_{1,7}$	$M_{1,8}$	$M_{1,9}$	$M_{1,10}$	$M_{1,11}$	$M_{1,12}$	$M_{1,13}$	$M_{1,14}$	$M_{1,15}$	$M_{1,16}$	$M_{1,17}$
$J_{1,1}$	39	0	0	0	0	0	0	0	0	0	0	36	0	32	0	23	0	34	38
$J_{2,1}$	0	0	0	0	0	36	0	0	0	0	0	0	0	0	0	34	0	7	8
$J_{3,1}$	0	0	12	0	0	0	0	0	0	0	0	24	0	0	0	0	0	37	32
$J_{4,1}$	38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	39	25
$J_{5,1}$	0	0	15	0	0	36	0	0	0	0	0	0	0	0	0	30	0	37	34
$J_{6,1}$	0	0	14	0	0	14	0	0	0	0	0	0	0	0	0	6	0	39	2
$J_{7,1}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$J_{8,1}$	29	0	0	0	0	0	0	0	0	0	0	18	0	0	0	10	0	24	9
$J_{9,1}$	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	27
$J_{10,1}$	0	0	38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	39
$J_{11,1}$	33	0	17	0	0	0	0	0	0	0	0	0	0	0	0	39	0	18	4

Change over from $J_{j'p'}$ to $J_{8,1}$ on $M_{i,k}$

	$M_{1,1}$	$M_{2,1}$	$M_{3,1}$	$M_{1,2}$	$M_{1,3}$	$M_{1,4}$	$M_{1,5}$	$M_{1,6}$	$M_{1,7}$	$M_{1,8}$	$M_{1,9}$	$M_{1,10}$	$M_{1,11}$	$M_{1,12}$	$M_{1,13}$	$M_{1,14}$	$M_{1,15}$	$M_{1,16}$	$M_{1,17}$
$J_{1,1}$	29	0	0	0	29	0	0	0	0	0	0	20	0	0	0	22	0	9	24
$J_{2.1}$	0	0	0	0	22	0	0	0	0	0	0	0	0	0	0	22	22	5	34
$J_{3,1}$	0	0	0	0	8	0	0	0	0	0	0	1	0	0	0	0	30	1	39
$J_{4.1}$	10	0	0	33	21	0	0	0	0	0	0	0	0	0	23	0	36	13	5
$J_{5,1}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	34	12	5	38
$J_{6.1}$	0	0	0	0	27	0	0	0	0	0	0	0	0	0	4	32	4	32	26
$J_{7,1}$	18	0	0	0	0	0	0	0	0	0	0	25	0	0	0	39	0	20	9
$J_{8,1}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$J_{9,1}$	0	0	0	33	9	0	0	0	0	0	0	0	0	0	0	0	16	19	31
$J_{10,1}$	0	0	0	0	3	0	0	0	0	0	0	0	0	0	38	0	16	11	33
$J_{11,1}$	7	0	0	0	6	0	0	0	0	0	0	0	0	0	24	7	17	9	5

Change over from $J_{j'p'}$ to $J_{9,1}$ on $M_{i,k}$

	$M_{1,1}$	$M_{2,1}$	$M_{3,1}$	$M_{1,2}$	$M_{1,3}$	$M_{1,4}$	$M_{1,5}$	$M_{1,6}$	$M_{1,7}$	$M_{1,8}$	$M_{1,9}$	$M_{1,10}$	$M_{1,11}$	$M_{1,12}$	$M_{1,13}$	$M_{1,14}$	$M_{1,15}$	$M_{1,16}$	$M_{1,17}$
$J_{1,1}$	0	3	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	33	21
$J_{2.1}$	0	12	0	0	28	0	0	0	0	0	38	0	0	0	0	0	4	8	33
$J_{3,1}$	0	0	20	0	36	0	0	0	0	0	24	0	0	0	0	0	13	29	28
$J_{4.1}$	0	0	0	9	22	0	0	0	0	0	0	0	0	0	0	0	9	24	17
$J_{5,1}$	0	37	19	0	0	0	0	0	0	0	0	0	0	0	0	0	5	22	23
$J_{6.1}$	0	0	8	0	14	0	0	0	0	0	0	0	0	0	0	0	36	29	8
$J_{7,1}$	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	7
$J_{8,1}$	0	0	0	32	35	0	0	0	0	0	0	0	0	0	0	0	28	27	37
$J_{9,1}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$J_{10,1}$	0	0	38	0	35	0	0	0	0	0	0	0	0	0	0	0	35	9	27
$J_{11,1}$	0	0	37	0	34	0	0	0	0	0	15	0	0	0	0	0	33	27	15

Change over from $J_{j'p'}$ to $J_{10,1}$ on $M_{i,k}$

	$M_{1,1}$	$M_{2,1}$	$M_{3,1}$	$M_{1,2}$	$M_{1,3}$	$M_{1,4}$	$M_{1,5}$	$M_{1,6}$	$M_{1,7}$	$M_{1,8}$	$M_{1,9}$	$M_{1,10}$	$M_{1,11}$	$M_{1,12}$	$M_{1,13}$	$M_{1,14}$	$M_{1,15}$	$M_{1,16}$	$M_{1,17}$
$J_{1,1}$	0	0	0	0	24	0	0	0	0	0	0	0	0	0	0	0	0	20	32
$J_{2.1}$	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	3	36	6
$J_{3,1}$	0	0	34	0	9	0	0	0	0	0	0	0	0	0	0	0	10	1	30
$J_{4.1}$	0	0	0	0	12	0	0	0	0	0	0	0	0	0	38	0	24	20	37
$J_{5,1}$	0	0	4	0	0	0	0	0	0	0	0	0	0	0	2	0	12	1	31
$J_{6.1}$	0	0	3	0	37	0	0	0	0	0	0	0	0	0	37	0	10	31	23
$J_{7,1}$	0	0	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	10
$J_{8,1}$	0	0	0	0	23	0	0	0	0	0	0	0	0	0	38	0	5	36	10
$J_{9,1}$	0	0	14	0	28	0	0	0	0	0	0	0	0	0	0	0	34	4	17
$J_{10,1}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$J_{11,1}$	0	0	30	0	12	0	0	0	13	0	0	0	0	0	19	0	22	18	36

Change over from $J_{j'p'}$ to $J_{11,1}$ on $M_{i,k}$																			
	$M_{1,1}$	$M_{2,1}$	$M_{3,1}$	$M_{1,2}$	$M_{1,3}$	$M_{1,4}$	$M_{1,5}$	$M_{1,6}$	$M_{1,7}$	$M_{1,8}$	$M_{1,9}$	$M_{1,10}$	$M_{1,11}$	$M_{1,12}$	$M_{1,13}$	$M_{1,14}$	$M_{1,15}$	$M_{1,16}$	$M_{1,17}$
$J_{1,1}$	17	0	0	0	33	0	0	0	0	0	0	0	0	0	0	1	0	31	14
$J_{2,1}$	0	0	0	0	20	0	0	0	0	0	6	0	0	0	0	26	35	26	14
$J_{3,1}$	0	0	6	0	18	0	0	0	0	0	34	0	0	0	0	0	33	22	19
$J_{4,1}$	3	0	0	0	31	0	0	0	0	0	0	0	0	0	8	0	24	34	8
$J_{5,1}$	0	0	28	0	0	0	0	0	0	0	0	0	0	0	25	25	35	27	6
$J_{6,1}$	0	0	15	0	7	0	0	0	0	0	0	0	0	0	16	26	27	2	29
$J_{7,1}$	24	0	24	0	0	0	0	0	0	0	0	0	0	0	0	3	0	4	13
$J_{8,1}$	8	0	0	0	38	0	0	0	0	0	0	0	0	0	37	20	4	10	39
$J_{9,1}$	0	0	14	0	38	0	0	0	0	0	8	0	0	0	0	0	6	29	3
$J_{10,1}$	0	0	19	0	25	0	0	0	18	0	0	0	0	0	28	0	38	3	25
$J_{11,1}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Change over reference to $J_{j'p'}$ on $M_{i,k}$

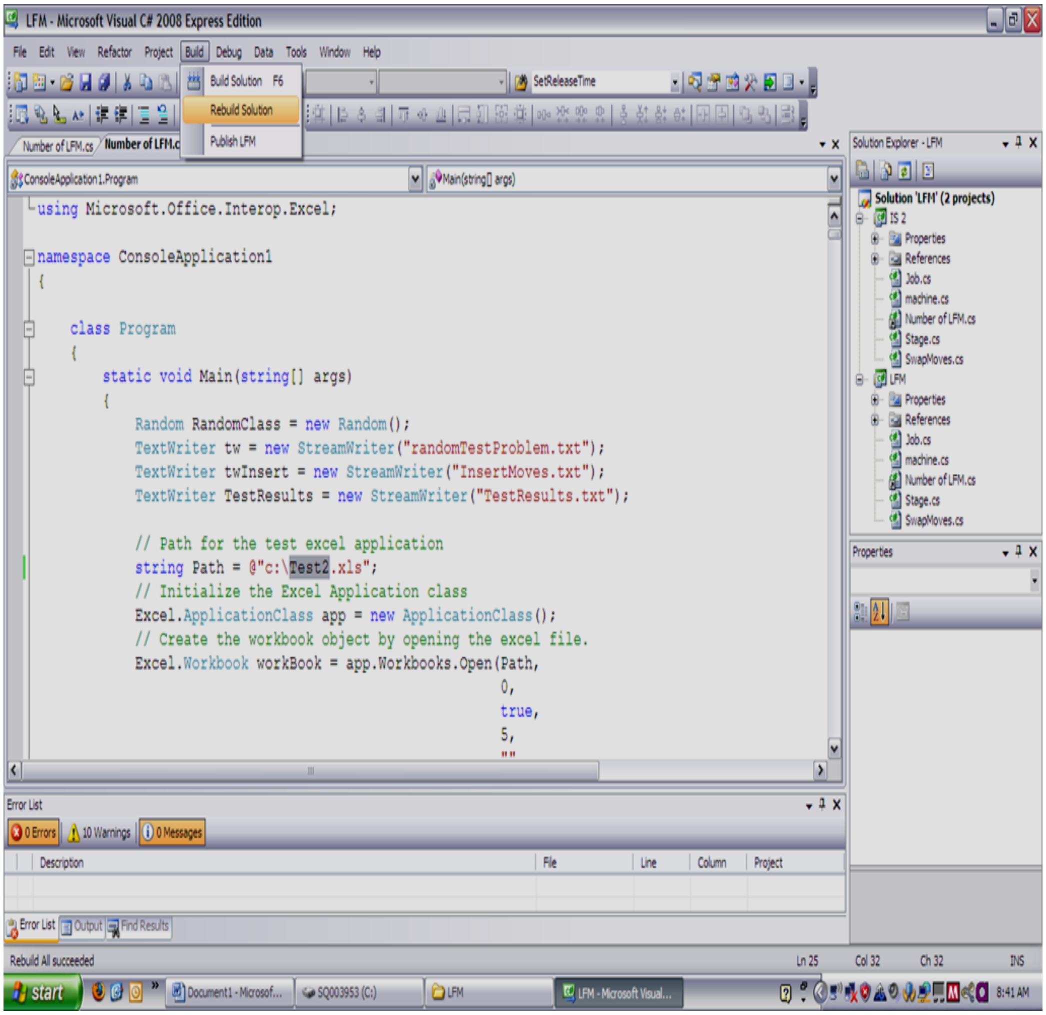
	$M_{1,1}$	$M_{2,1}$	$M_{3,1}$	$M_{1,2}$	$M_{1,3}$	$M_{1,4}$	$M_{1,5}$	$M_{1,6}$	$M_{1,7}$	$M_{1,8}$	$M_{1,9}$	$M_{1,10}$	$M_{1,11}$	$M_{1,12}$	$M_{1,13}$	$M_{1,14}$	$M_{1,15}$	$M_{1,16}$	$M_{1,17}$
$J_{1,1}$	33	26	0	0	18	0	0	0	0	0	0	10	0	34	0	2	0	26	4
$J_{2,1}$	0	17	0	0	19	14	0	0	0	0	24	0	0	0	0	12	4	38	34
$J_{3,1}$	0	0	7	0	3	0	0	0	0	14	30	28	0	0	0	0	19	22	2
$J_{4,1}$	19	0	0	5	2	0	0	0	0	0	0	0	12	0	1	0	13	15	3
$J_{5,1}$	0	28	22	0	0	1	12	0	0	0	0	0	0	0	3	20	12	25	26
$J_{6,1}$	0	0	36	0	7	24	0	0	0	0	0	0	0	0	16	13	13	37	13
$J_{7,1}$	39	0	29	0	0	14	0	0	0	0	0	23	0	36	0	9	0	13	31
$J_{8,1}$	15	0	0	9	32	0	0	0	0	0	0	17	0	0	39	25	22	16	32
$J_{9,1}$	0	23	2	30	19	0	0	0	0	0	39	0	0	0	0	0	27	28	35
$J_{10,1}$	0	0	9	0	5	0	0	0	20	0	0	0	0	0	22	0	8	14	35
$J_{11,1}$	32	0	22	0	3	0	0	0	37	0	7	0	0	0	27	18	37	29	3

APPENDIX B.

User Manual for Scheduling software

Procedure for running the scheduling algorithm

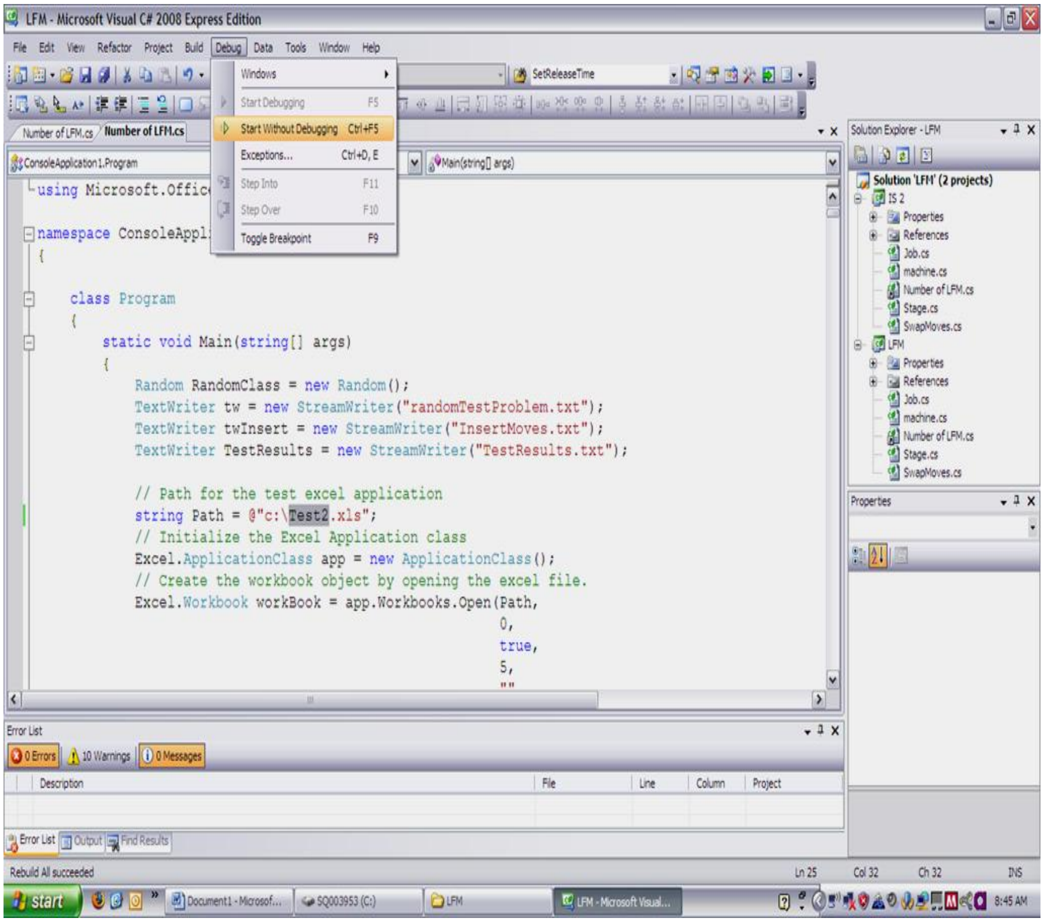
1) Create five new excel files namely IS1, IS2, IS3, IS4 and IS5 and place these in the C:\ drive. Each file will contain the final best solution obtained from a particular initial solution. For example, IS1 will contain best solution for a given problem, using Initial Solution 1 as a starting point.
2) Place the excel data file (which has appropriate data for job release time, due date, weight, setup time, etc.) in the C:\ drive.
3) Save this excel data file as “Test.xls”.
4) If we need to provide the algorithm with a new problem instance, revise the same “Test.xls” with new data set.
5) Unzip the “TestProduct” folder and double click on it. Next→ double click on the folder named “Combine” and run the C# file which is also named “Combine”
6) Click on the “**Build**” Tab and select “**Rebuild Solution**”.



```
using Microsoft.Office.Interop.Excel;

namespace ConsoleApplication1
{

    class Program
    {
        static void Main(string[] args)
        {
            Random RandomClass = new Random();
            TextWriter tw = new StreamWriter("randomTestProblem.txt");
            TextWriter twInsert = new StreamWriter("InsertMoves.txt");
            TextWriter TestResults = new StreamWriter("TestResults.txt");

            // Path for the test excel application
            string Path = @"c:\Test2.xls";
            // Initialize the Excel Application class
            Excel.ApplicationClass app = new ApplicationClass();
            // Create the workbook object by opening the excel file.
            Excel.Workbook workBook = app.Workbooks.Open(Path,
                                                         0,
                                                         true,
                                                         5,
                                                         ""
```

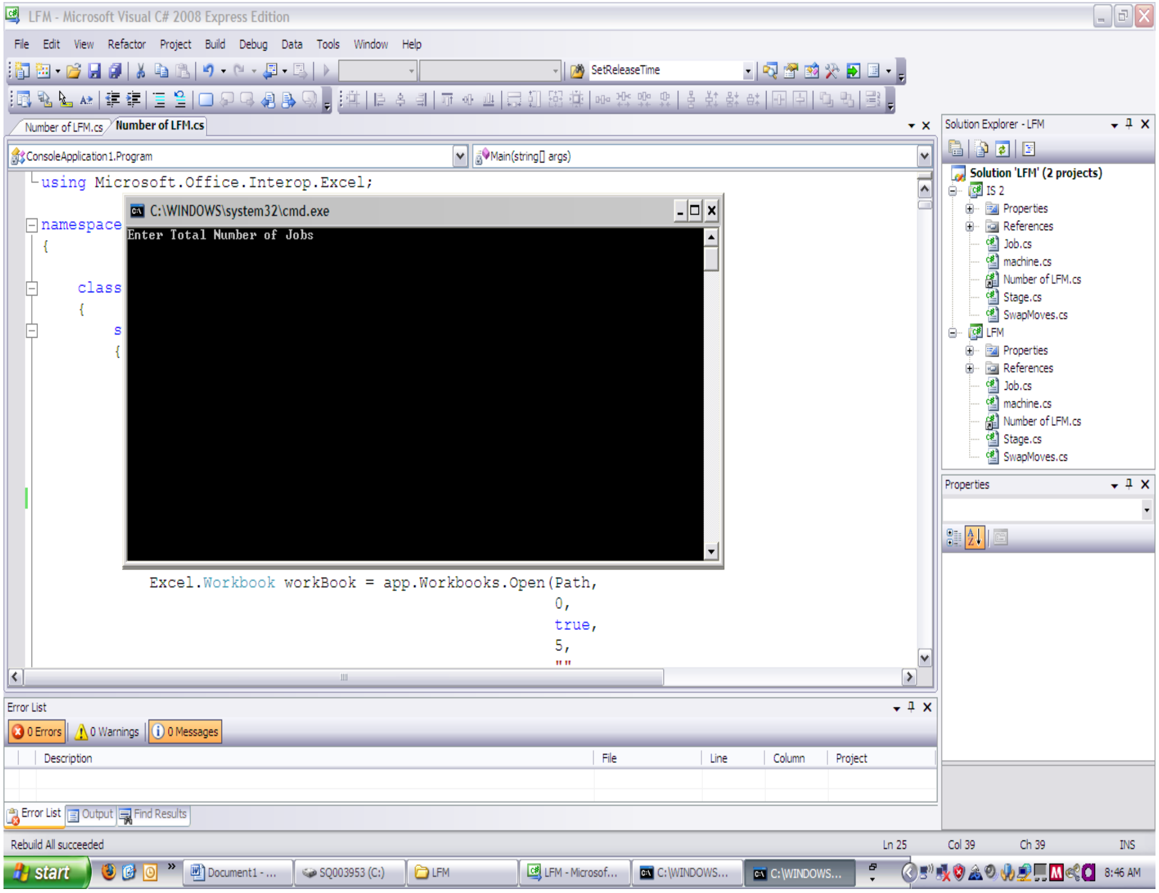
Rebuild All succeeded

7) Now click on the “**Debug**” tab and select “**Start without Debugging**”.



```
using Microsoft.Offic

namespace ConsoleAppl
{

    class Program
    {
        static void Main(string[] args)
        {
            Random RandomClass = new Random();
            TextWriter tw = new StreamWriter("randomTestProblem.txt");
            TextWriter twInsert = new StreamWriter("InsertMoves.txt");
            TextWriter TestResults = new StreamWriter("TestResults.txt");

            // Path for the test excel application
            string Path = @"c:\Test2.xls";
            // Initialize the Excel Application class
            Excel.ApplicationClass app = new ApplicationClass();
            // Create the workbook object by opening the excel file.
            Excel.Workbook workBook = app.Workbooks.Open(Path,
                                                         0,
                                                         true,
                                                         5,
                                                         ""
```

8) Five console windows will pop up. Each window will prompt the user to enter the total number of jobs. Enter the total number of jobs in each console window (that the new problem instance has). For example if the new problem instance has a total of 50 jobs, then simply enter "**50**" using the keyboard and press "**Enter**" key.



```
using Microsoft.Office.Interop.Excel;

namespace
{

    class
    {
        s
        {
```

```
Enter Total Number of Jobs
```

```
            Excel.Workbook workBook = app.Workbooks.Open(Path,
                                                          0,
                                                          true,
                                                          5,
                                                          ""
```

9) The algorithm's execution will begin immediately and after few seconds, five excel sheets will appear on the screen, showing the best results obtained using each initial solution.

10) The first row in each excel sheet will read as follows:

The best **TWT** (total weighted tardiness) obtained from **Initial Solution** 1 (or 2 or 3) is ______ **(a value).**

11) Among the five different excel spreadsheets (generated by the algorithm), choose the excel file that has the least **TWT** (total weighted tardiness) and save it. Discard the other excel files.

Understanding the final schedule

1) The final schedule provides us information regarding which job should be processed on which particular machine (to obtain a near optimal solution). It also gives us the **time** when the **setup of a job** should start, along with the **time** when the actual **run of the job** should start. Completion **time of each job** (**in each stage**) is also provided to avoid any confusion.
2) Additional Clarifications:
    - **Setup start time**: The setup start time presented in the schedule is the **latest time** by which the setup of a job must begin on a particular machine.
    - **Start time of the run:** The start time of the run of a job on a particular machine is the time when the setup of a job is finished and the actual run of a job should begin.
    - **Completion time:** Completion time of a job on a particular machine is **the start time of the run + run time of that job**.
    - In the final schedule, jobs are sorted in increasing order (for each stage) according **"the start times of their runs"**.

APPENDIX C

CMAX EVALUATION

Proj	Job	Runtime on Machine	Machine Availability	Beta	Avg set up	M/c Avai + beta*Avg set up	Job release time	Add runtime	divide by num of m/c	Sum total	Sum total divided by 3
1	1	45	3	0.52	24.40	15.70	2	60.70	53.25	574.66	191.55
1	1	35	8	0.24	11.50	10.80	2	45.80			
1	2	34	8	0.51	23.00	19.63	4	53.63	53.63		
1	3	33	4	0.34	8.85	6.99	4	39.99	39.99		
1	4	34	3	0.47	19.60	12.14	8	46.14	46.14		
1	5	41	8	0.50	16.00	16.01	4	57.01	52.41		
1	5	32	4	0.59	20.00	15.80	4	47.80			
1	6	43	4	0.53	22.71	16.07	6	59.07	59.07		
1	7	40	3	0.81	35.60	31.72	8	71.72	65.61		
1	7	45	4	0.52	20.14	14.50	8	59.50			
1	8	37	3	0.47	15.80	10.44	7	47.44	47.44		
1	9	24	8	0.27	18.75	13.15	4	37.15	46.75		
1	9	47	4	0.29	18.71	9.35	4	56.35			
1	10	46	4	0.27	15.43	8.16	4	54.16	54.16		
1	11	48	3	0.34	16.80	8.75	4	56.75	56.21		
1	11	41	4	0.58	18.29	14.68	4	55.68			

Hybrid Critical Ratio Evaluations

P	J	Wt	Release	DueDate	M1	M2	M3	Avg Set up on M1	Avg Set up on M2	Avg Set up on M3	HCR M1	HCR M2	HCR M3
					3	8	4						
1	1	1	2	1684	45	35	0	24.40	11.75	0.00	24.27	36.02	
1	2	2	4	520	0	34	0	0.00	23.00	0.00		4.56	
1	3	1	4	339	0	0	33	0.00	0.00	8.86			8.10
1	4	3	8	1462	34	0	0	19.60	0.00	0.00	9.09		
1	5	3	4	1501	0	41	32	0.00	16.00	20.00		8.78	9.62
1	6	2	6	517	0	0	43	0.00	0.00	22.71			3.93
1	7	2	8	388	40	0	45	35.60	0.00	20.14	2.57		2.98
1	8	3	7	603	37	0	0	15.80	0.00	0.00	3.81		
1	9	2	4	441	0	24	47	0.00	18.75	18.71		5.16	3.36
1	10	1	4	819	0	0	46	0.00	0.00	15.43			13.33
1	11	1	4	350	48	0	41	16.80	0.00	18.29	5.40		5.90

Appendix D

Nine job block 1

P	**J**	**W**	**RR**	**DD**	**$M_{1,1}$**	**$M_{2,1}$**	**$M_{3,1}$**	**$M_{1,2}$**	**$M_{1,3}$**	**$M_{1,4}$**	**$M_{1,5}$**	**$M_{1,6}$**	**$M_{1,7}$**	**$M_{1,8}$**	**$M_{1,9}$**	**$M_{1,10}$**
					2	5	7	21	26	30	32	37	44	47	53	59
Runtime of Jobs																
1	1	2	6	2593	45	33	0	40	40	0	0	0	42	29	40	0
1	2	1	4	1063	0	0	41	0	27	0	0	0	0	43	0	0
1	3	2	2	1664	38	31	0	0	32	0	0	0	0	0	0	34
1	4	2	3	2435	0	0	31	0	30	0	0	0	0	0	45	0
1	5	2	4	2821	0	0	38	0	0	0	31	35	0	0	35	26
1	6	3	8	2542	0	41	0	0	37	0	0	0	0	0	40	0
1	7	1	8	2732	0	0	40	0	34	29	0	0	32	44	37	42
1	8	1	7	2582	47	0	0	38	39	0	37	33	0	37	33	0
1	9	3	2	1171	38	0	0	0	39	0	0	0	0	0	37	0

P	**J**	**W**	**RR**	**DD**	**$M_{1,11}$**	**$M_{1,12}$**	**$M_{1,13}$**	**$M_{1,14}$**	**$M_{1,15}$**	**$M_{1,16}$**	**$M_{1,17}$**
					61	64	68	70	77	81	86
Runtime of Jobs											
1	1	2	6	2593	0	0	40	29	35	27	31
1	2	1	4	1063	0	39	0	0	0	36	34
1	3	2	2	1664	41	0	0	26	28	29	40
1	4	2	3	2435	0	36	0	41	36	27	44
1	5	2	4	2821	0	33	0	43	28	40	38
1	6	3	8	2542	0	35	30	0	0	26	37
1	7	1	8	2732	0	34	35	28	0	38	39
1	8	1	7	2582	0	0	40	29	37	37	47
1	9	3	2	1171	0	0	0	40	28	25	47

Nine job block 2

P	**J**	**W**	**RR**	**DD**	**$M_{1,1}$**	**$M_{2,1}$**	**$M_{3,1}$**	**$M_{1,2}$**	**$M_{1,3}$**	**$M_{1,4}$**	**$M_{1,5}$**	**$M_{1,6}$**	**$M_{1,7}$**	**$M_{1,8}$**	**$M_{1,9}$**	**$M_{1,10}$**
					4	5	8	25	33	40	47	55	59	66	74	79
Runtime of Jobs																
1	1	3	7	964	0	0	28	0	41	0	0	0	0	0	0	0
1	2	2	6	735	47	0	0	0	36	0	0	0	0	0	0	0
1	3	2	6	1643	0	30	0	0	29	31	0	0	46	45	0	41
1	4	1	3	1702	0	0	28	34	0	0	31	0	0	0	0	0
1	5	1	3	1067	0	34	0	38	23	0	0	0	0	0	46	0
1	6	3	5	1365	42	0	0	41	26	31	0	0	0	34	0	0
1	7	3	5	1326	0	41	0	0	25	0	0	0	0	35	0	0
1	8	2	5	836	37	0	37	0	34	0	0	0	0	29	35	28
1	9	1	6	1092	0	32	0	0	25	0	42	0	31	41	48	0

P	**J**	**W**	**RR**	**DD**	**$M_{1,11}$**	**$M_{1,12}$**	**$M_{1,13}$**	**$M_{1,14}$**	**$M_{1,15}$**	**$M_{1,16}$**	**$M_{1,17}$**
					85	93	98	106	111	119	121
Runtime of Jobs											
1	1	3	7	964	0	27	0	0	29	42	45
1	2	2	6	735	0	0	0	0	0	38	38
1	3	2	6	1643	0	0	0	34	41	41	42
1	4	1	3	1702	0	0	28	39	44	33	32
1	5	1	3	1067	0	0	0	35	39	30	44
1	6	3	5	1365	0	0	0	27	38	25	43
1	7	3	5	1326	0	32	0	0	45	37	43
1	8	2	5	836	0	30	27	0	40	25	34
1	9	1	6	1092	0	0	0	34	42	39	46

Nine job block 3

P	J	W	RR	DD	$M_{1,1}$	$M_{2,1}$	$M_{3,1}$	$M_{1,2}$	$M_{1,3}$	$M_{1,4}$	$M_{1,5}$	$M_{1,6}$	$M_{1,7}$	$M_{1,8}$	$M_{1,9}$	$M_{1,10}$
					8	6	4	25	28	34	42	44	49	55	59	64
Runtime of Jobs																
1	1	3	3	498	0	25	0	0	43	28	0	47	0	0	0	26
1	2	3	3	500	0	27	0	0	0	0	0	0	0	0	40	0
1	3	1	6	966	30	0	0	0	30	39	0	0	0	0	25	0
1	4	2	8	523	0	36	0	33	43	0	0	0	0	0	0	30
1	5	1	4	508	39	38	0	0	38	0	0	0	31	0	30	0
1	6	3	6	543	0	0	31	28	41	0	32	31	0	30	0	0
1	7	2	6	822	25	0	37	0	45	0	0	0	0	0	0	0
1	8	2	5	527	27	0	24	0	43	0	0	0	0	42	29	35
1	9	1	4	476	33	0	0	28	43	31	0	0	30	0	37	0

P	J	W	RR	DD	$M_{1,11}$	$M_{1,12}$	$M_{1,13}$	$M_{1,14}$	$M_{1,15}$	$M_{1,16}$	$M_{1,17}$
					68	70	73	75	81	84	87
Runtime of Jobs											
1	1	3	3	498	0	39	36	0	40	43	41
1	2	3	3	500	0	33	28	33	43	34	27
1	3	1	6	966	0	0	0	33	45	31	38
1	4	2	8	523	40	0	0	41	0	27	32
1	5	1	4	508	0	30	38	0	0	43	28
1	6	3	6	543	0	0	0	0	47	32	31
1	7	2	6	822	0	32	0	37	33	36	32
1	8	2	5	527	0	36	0	41	39	43	38
1	9	1	4	476	0	0	35	30	44	28	36

Nine job block 5

P	J	W	RR	DD	$M_{1,1}$	$M_{2,1}$	$M_{3,1}$	$M_{1,2}$	$M_{1,3}$	$M_{1,4}$	$M_{1,5}$	$M_{1,6}$	$M_{1,7}$	$M_{1,8}$	$M_{1,9}$	$M_{1,10}$
					2	4	3	14	21	26	33	35	38	42	47	54
Runtime of Jobs																
1	1	1	8	225	36	25	0	39	37	0	0	0	0	0	24	0
1	2	1	8	192	0	42	0	47	0	27	31	0	0	0	0	0
1	3	2	4	1996	0	0	33	40	0	0	0	0	0	42	0	0
1	4	2	6	497	33	43	0	0	46	0	0	0	0	43	0	37
1	5	3	4	396	0	42	0	0	33	0	0	0	0	28	0	0
1	6	1	3	299	33	32	0	0	38	0	0	0	0	25	0	0
1	7	3	8	345	0	28	40	0	34	0	0	0	0	0	39	0
1	8	1	6	272	0	31	0	0	32	40	0	0	0	33	24	0
1	9	3	3	249	30	0	0	37	33	0	0	0	0	40	41	0

P	J	W	RR	DD	$M_{1,11}$	$M_{1,12}$	$M_{1,13}$	$M_{1,14}$	$M_{1,15}$	$M_{1,16}$	$M_{1,17}$
					57	62	68	74	81	84	89
Runtime of Jobs											
1	1	1	8	225	0	27	39	31	0	41	41
1	2	1	8	192	0	0	36	31	38	33	33
1	3	2	4	1996	0	0	0	0	34	35	31
1	4	2	6	497	0	0	41	39	0	39	43
1	5	3	4	396	0	0	28	41	34	36	31
1	6	1	3	299	0	0	0	37	47	37	28
1	7	3	8	345	38	0	32	0	37	35	34
1	8	1	6	272	0	0	0	41	40	37	35
1	9	3	3	249	0	34	32	0	47	31	29

TWELVE JOBS BLOCK 1

P	**J**	**W**	**RR**	**DD**	**$M_{1,1}$**	**$M_{2,1}$**	**$M_{3,1}$**	**$M_{1,2}$**	**$M_{1,3}$**	**$M_{1,4}$**	**$M_{1,5}$**	**$M_{1,6}$**	**$M_{1,7}$**	**$M_{1,8}$**	**$M_{1,9}$**	**$M_{1,10}$**
					2	6	7	19	26	34	41	43	48	51	57	61
Runtime of Jobs																
1	1	3	2	2344	37	0	32	0	32	34	0	0	0	39	45	0
1	2	2	5	3180	0	0	30	35	28	0	0	0	0	0	0	0
1	3	2	8	3103	0	0	47	0	34	0	0	0	48	0	0	29
1	4	2	2	2253	0	36	36	38	0	0	0	0	0	0	42	35
1	5	1	8	995	0	39	0	0	0	26	0	0	0	0	39	0
1	6	3	2	3598	0	28	33	31	38	0	0	0	31	0	0	0
1	7	2	2	3052	0	0	46	0	37	34	0	0	0	0	38	36
1	8	1	2	3185	0	26	31	0	41	26	24	0	0	0	28	0
1	9	1	4	3376	0	41	0	0	0	0	0	0	0	0	35	42
1	10	1	6	3178	41	0	0	0	30	0	0	0	0	42	0	0
1	11	1	5	1431	35	0	45	0	37	0	0	0	0	0	0	40
1	12	1	4	3498	29	33	0	0	27	0	0	0	0	0	0	0

P	**J**	**W**	**RR**	**DD**	**$M_{1,11}$**	**$M_{1,12}$**	**$M_{1,13}$**	**$M_{1,14}$**	**$M_{1,15}$**	**$M_{1,16}$**	**$M_{1,17}$**
					64	67	74	77	82	90	96
Runtime of Jobs											
1	1	3	2	2344	0	0	39	37	0	41	42
1	2	2	5	3180	0	0	33	29	0	39	29
1	3	2	8	3103	0	0	0	0	44	27	39
1	4	2	2	2253	0	35	36	42	0	42	35
1	5	1	8	995	0	45	47	36	28	26	44
1	6	3	2	3598	0	0	0	0	32	39	29
1	7	2	2	3052	0	42	45	35	37	39	30
1	8	1	2	3185	0	32	32	30	0	37	40
1	9	1	4	3376	0	43	0	36	0	27	28
1	10	1	6	3178	0	0	0	29	42	38	35
1	11	1	5	1431	0	42	40	27	38	43	41

1	12	1	4	3498	0	30	0	27	28	37	29

TWELVE JOBS BLOCK 2

P	**J**	**W**	**RR**	**DD**	$M_{1,1}$	$M_{2,1}$	$M_{3,1}$	$M_{1,2}$	$M_{1,3}$	$M_{1,4}$	$M_{1,5}$	$M_{1,6}$	$M_{1,7}$	$M_{1,8}$	$M_{1,9}$	$M_{1,10}$
					4	8	5	24	29	31	39	42	50	58	66	71
Runtime of Jobs																
1	1	1	3	1540	0	33	0	0	31	0	0	0	0	26	0	0
1	2	3	8	1967	44	0	0	42	32	0	0	24	0	0	0	0
1	3	3	7	2854	0	32	0	0	40	29	34	0	23	0	33	0
1	4	3	8	2680	0	37	31	0	26	0	45	0	0	0	0	30
1	5	3	8	2947	0	35	0	35	34	28	0	0	0	0	0	0
1	6	2	6	2169	0	0	38	0	34	0	0	0	31	36	27	0
1	7	3	5	2461	0	0	33	0	31	33	30	0	0	33	31	0
1	8	3	4	2798	30	40	0	0	32	0	0	0	0	43	0	23
1	9	2	3	2043	44	0	0	0	26	0	0	0	0	0	32	0
1	10	2	5	2088	0	0	41	0	42	0	33	0	0	0	0	0
1	11	1	6	1294	42	27	0	42	36	43	0	0	0	0	0	0
1	12	2	8	1098	0	37	0	38	36	0	0	0	27	0	0	26

P	**J**	**W**	**RR**	**DD**	$M_{1,11}$	$M_{1,12}$	$M_{1,13}$	$M_{1,14}$	$M_{1,15}$	$M_{1,16}$	$M_{1,17}$
					76	84	86	94	96	102	106
Runtime of Jobs											
1	1	1	3	1540	0	0	0	47	29	46	39
1	2	3	8	1967	0	37	33	44	30	40	40
1	3	3	7	2854	0	0	44	45	38	40	48
1	4	3	8	2680	28	30	0	44	29	36	41
1	5	3	8	2947	0	0	0	41	38	36	35
1	6	2	6	2169	0	0	0	43	37	36	32
1	7	3	5	2461	0	0	31	35	41	44	47
1	8	3	4	2798	0	45	0	45	36	34	37
1	9	2	3	2043	39	28	33	48	39	31	32
1	10	2	5	2088	0	33	0	32	47	36	30
1	11	1	6	1294	0	0	0	32	43	34	34
1	12	2	8	1098	0	0	39	31	0	37	41

TWELVE BLOCKS JOB 3

P	J	W	RR	DD	$M_{1,1}$	$M_{2,1}$	$M_{3,1}$	$M_{1,2}$	$M_{1,3}$	$M_{1,4}$	$M_{1,5}$	$M_{1,6}$	$M_{1,7}$	$M_{1,8}$	$M_{1,9}$	$M_{1,10}$
					5	8	7	23	26	30	32	35	42	50	58	65
Runtime of Jobs																
1	1	3	5	659	26	30	0	0	27	0	0	0	0	0	0	39
1	2	2	6	634	41	35	0	0	38	0	0	0	33	25	0	0
1	3	3	7	676	0	31	37	36	0	37	0	0	0	0	0	0
1	4	1	2	567	0	36	0	30	31	0	0	0	29	32	31	27
1	5	2	8	665	0	0	34	0	31	0	0	0	0	0	0	25
1	6	3	4	624	0	38	0	0	0	38	0	0	31	0	26	35
1	7	1	3	636	37	0	0	47	37	34	0	41	0	0	0	0
1	8	2	4	999	30	0	0	0	36	27	0	0	0	38	0	36
1	9	1	4	675	0	44	0	0	32	0	0	0	0	0	0	25
1	10	1	5	597	0	0	41	0	35	42	0	0	0	0	0	0
1	11	2	4	617	0	35	0	33	30	0	0	0	29	0	0	0
1	12	3	7	646	33	0	0	0	28	0	0	0	29	0	0	0

P	J	W	RR	DD	$M_{1,11}$	$M_{1,12}$	$M_{1,13}$	$M_{1,14}$	$M_{1,15}$	$M_{1,16}$	$M_{1,17}$
					73	77	80	82	89	96	101
Runtime of Jobs											
1	1	3	5	659	0	0	35	30	0	41	33
1	2	2	6	634	0	33	40	0	35	28	23
1	3	3	7	676	40	0	0	0	0	45	40
1	4	1	2	567	0	0	36	30	0	41	30
1	5	2	8	665	0	32	0	38	34	31	28
1	6	3	4	624	0	0	0	35	33	42	25
1	7	1	3	636	0	0	0	35	33	27	26
1	8	2	4	999	0	35	45	38	37	30	40
1	9	1	4	675	0	0	0	25	31	35	31
1	10	1	5	597	0	0	0	31	44	34	22
1	11	2	4	617	0	0	0	27	0	35	29

1	12	3	7	646	0	0	37	29	43	32	29

TWELVE JOBS BLOCK 5

P	**J**	**W**	**RR**	**DD**	**$M_{1,1}$**	**$M_{2,1}$**	**$M_{3,1}$**	**$M_{1,2}$**	**$M_{1,3}$**	**$M_{1,4}$**	**$M_{1,5}$**	**$M_{1,6}$**	**$M_{1,7}$**	**$M_{1,8}$**	**$M_{1,9}$**	**$M_{1,10}$**
					4	4	4	17	21	24	26	30	32	40	44	50
Runtime of Jobs																
1	1	1	7	635	0	26	36	38	28	0	0	0	0	31	0	0
1	2	3	8	554	0	30	0	0	39	0	31	0	0	0	0	0
1	3	1	2	254	0	0	42	0	28	0	38	0	0	0	38	40
1	4	2	2	236	40	0	30	25	0	0	0	0	0	0	29	0
1	5	3	4	1006	26	0	0	30	24	0	0	43	0	0	0	0
1	6	3	7	538	0	0	40	39	38	38	0	0	0	0	0	0
1	7	2	6	563	24	0	0	24	38	0	0	0	0	34	0	0
1	8	3	6	370	0	26	0	0	25	0	0	0	0	0	0	0
1	9	3	4	698	0	0	28	0	40	0	0	0	0	0	31	0
1	10	3	6	1809	37	40	0	31	30	39	0	0	0	32	28	42
1	11	3	6	309	29	0	0	0	34	24	30	0	0	0	0	41
1	12	1	2	297	0	32	0	22	42	35	0	0	0	0	31	0

P	**J**	**W**	**RR**	**DD**	$\mathbf{M_{1,11}}$	$\mathbf{M_{1,12}}$	$\mathbf{M_{1,13}}$	$\mathbf{M_{1,14}}$	$\mathbf{M_{1,15}}$	$\mathbf{M_{1,16}}$	$\mathbf{M_{1,17}}$
					54	58	66	70	75	82	89
Runtime of Jobs											
1	1	1	7	635	0	36	0	0	35	37	36
1	2	3	8	554	0	47	33	37	27	37	43
1	3	1	2	254	0	0	27	27	38	35	35
1	4	2	2	236	0	32	0	35	0	40	33
1	5	3	4	1006	0	0	0	40	37	34	34
1	6	3	7	538	0	0	33	30	37	25	43
1	7	2	6	563	0	0	0	29	28	39	35
1	8	3	6	370	0	0	0	0	0	38	48
1	9	3	4	698	35	0	0	0	27	38	41
1	10	3	6	1809	0	0	39	29	0	29	38
1	11	3	6	309	0	36	34	34	45	37	30
1	12	1	2	297	0	0	40	0	37	38	30

SEVENTEEN JOBS BLOCK 1

P	**J**	**W**	**R**	**DD**	**$M_{1,1}$**	**$M_{2,1}$**	**$M_{3,1}$**	**$M_{1,2}$**	**$M_{1,3}$**	**$M_{1,4}$**	**$M_{1,5}$**	**$M_{1,6}$**	**$M_{1,7}$**	**$M_{1,8}$**	**$M_{1,9}$**	**$M_{1,10}$**
					4	6	7	20	24	27	34	42	46	52	57	59
Runtime of Jobs																
1	1	3	7	4381	34	0	37	32	29	0	0	0	39	40	0	0
1	2	1	4	4197	0	0	42	0	28	0	0	0	33	29	35	0
1	3	2	8	4641	0	25	0	0	29	0	0	0	24	0	0	34
1	4	2	5	3369	37	0	0	0	38	0	0	0	0	33	0	0
1	5	3	8	4640	0	0	28	0	26	0	0	0	0	34	0	0
1	6	2	5	4347	0	0	29	0	29	0	0	0	0	24	0	33
1	7	2	6	3983	0	0	42	27	27	0	0	0	0	0	28	0
1	8	2	7	4249	0	34	0	0	26	0	0	0	33	29	0	36
1	9	1	2	4651	0	0	40	0	33	0	0	0	0	36	0	0
1	10	2	2	4480	32	25	0	0	41	0	0	0	0	31	0	28
2	1	1	3	4675	30	0	41	0	31	0	0	0	0	0	0	0
2	2	3	8	4487	0	0	26	0	0	0	0	0	32	27	0	0
2	3	1	6	4124	0	29	0	0	26	0	0	0	26	0	0	0
2	4	2	7	3965	0	28	0	37	41	34	0	0	37	29	30	0
2	5	1	8	4511	31	0	0	0	32	0	0	0	0	23	31	0
2	6	2	3	4561	0	0	25	33	36	0	0	0	30	0	0	0
2	7	1	6	1594	0	29	0	30	29	0	41	0	0	0	0	30

P	**J**	**W**	**R**	**DD**	$M_{1,11}$	$M_{1,12}$	$M_{1,13}$	$M_{1,14}$	$M_{1,15}$	$M_{1,16}$	$M_{1,17}$
					61	66	73	76	81	87	91
Runtime of Jobs											
1	1	3	7	4381	0	0	32	39	32	37	32
1	2	1	4	4197	0	0	0	39	42	31	38
1	3	2	8	4641	36	38	32	36	39	29	27
1	4	2	5	3369	0	33	34	40	0	34	32
1	5	3	8	4640	0	0	36	0	0	34	37
1	6	2	5	4347	0	0	0	0	29	33	36
1	7	2	6	3983	29	0	0	30	29	27	31
1	8	2	7	4249	0	0	32	43	0	28	23
1	9	1	2	4651	0	47	47	0	38	39	40
1	10	2	2	4480	0	0	39	39	27	26	28
2	1	1	3	4675	0	32	41	43	24	33	28
2	2	3	8	4487	0	0	37	0	30	25	28
2	3	1	6	4124	0	0	0	0	27	33	39
2	4	2	7	3965	31	0	0	42	26	28	38
2	5	1	8	4511	37	38	0	31	0	39	24
2	6	2	3	4561	0	0	0	0	32	30	38
2	7	1	6	1594	0	0	34	0	0	24	37

SEVENTEEN JOBS BLOCK 2

P	**J**	**W**	**RR**	**DD**	$M_{1,1}$	$M_{2,1}$	$M_{3,1}$	$M_{1,2}$	$M_{1,3}$	$M_{1,4}$	$M_{1,5}$	$M_{1,6}$	$M_{1,7}$	$M_{1,8}$	$M_{1,9}$	$M_{1,10}$
					4	8	4	22	28	35	39	44	52	54	61	67
Runtime of Jobs																
1	1	2	6	3052	38	0	0	33	37	41	0	0	0	27	39	0
1	2	3	3	1475	0	0	33	0	41	0	0	0	0	0	0	29
1	3	1	5	1676	32	0	0	41	43	0	0	0	0	35	40	0
1	4	1	5	1703	0	28	0	0	27	0	0	44	0	0	0	37
1	5	2	2	1290	45	0	0	0	29	0	34	0	0	0	0	0
1	6	1	4	2657	0	0	38	36	27	26	0	0	27	30	0	0
1	7	2	3	1784	42	0	0	0	28	0	31	0	0	0	33	0
1	8	1	4	1879	0	0	32	0	27	0	0	45	0	30	0	34
1	9	2	5	3575	34	29	0	27	26	0	42	0	0	0	0	0
1	10	3	6	2971	38	30	0	0	32	41	0	0	0	0	28	0
2	1	1	7	2857	0	0	31	29	38	38	0	0	0	0	41	30
2	2	1	2	1685	36	0	0	0	42	35	0	0	0	0	24	0
2	3	1	2	1360	0	0	42	31	37	0	0	0	39	0	38	0
2	4	1	4	1490	0	0	27	24	36	0	0	0	0	40	40	0
2	5	2	6	1447	0	39	0	0	35	0	0	0	26	0	24	0
2	6	2	8	2002	0	0	32	0	26	0	0	0	0	0	0	0
2	7	3	5	2238	0	44	0	25	44	32	38	0	0	0	0	0

P	**J**	**W**	**RR**	**DD**	**$M_{1,11}$**	**$M_{1,12}$**	**$M_{1,13}$**	**$M_{1,14}$**	**$M_{1,15}$**	**$M_{1,16}$**	**$M_{1,17}$**
					73	76	81	88	93	96	104
Runtime of Jobs											
1	1	2	6	3052	0	0	0	39	43	31	29
1	2	3	3	1475	0	0	45	41	33	40	41
1	3	1	5	1676	0	0	37	37	31	48	29
1	4	1	5	1703	41	33	33	35	0	33	44
1	5	2	2	1290	0	42	0	0	40	47	32
1	6	1	4	2657	0	29	0	33	0	34	31
1	7	2	3	1784	0	45	0	23	0	44	37
1	8	1	4	1879	0	43	0	37	30	48	36
1	9	2	5	3575	0	46	0	0	43	39	39
1	10	3	6	2971	0	37	40	40	42	40	40
2	1	1	7	2857	0	38	0	36	40	33	38
2	2	1	2	1685	0	0	0	25	32	36	42
2	3	1	2	1360	0	30	0	0	28	45	44
2	4	1	4	1490	0	0	45	37	30	32	37
2	5	2	6	1447	0	41	0	24	45	34	33
2	6	2	8	2002	0	0	0	0	32	47	38
2	7	3	5	2238	0	32	42	34	41	30	39

SEVENTEEN JOBS BLOCK 3

P	**J**	**W**	**RR**	**DD**	**$M_{1,1}$**	**$M_{2,1}$**	**$M_{3,1}$**	**$M_{1,2}$**	**$M_{1,3}$**	**$M_{1,4}$**	**$M_{1,5}$**	**$M_{1,6}$**	**$M_{1,7}$**	**$M_{1,8}$**	**$M_{1,9}$**	**$M_{1,10}$**
					5	2	8	18	24	31	39	45	51	55	61	66
Runtime of Jobs																
1	1	1	2	1049	0	23	0	45	34	0	39	0	0	0	0	0
1	2	1	6	930	24	31	0	0	42	41	0	0	38	0	0	0
1	3	1	7	1022	0	0	38	0	0	0	0	26	0	0	0	0
1	4	3	6	940	35	0	0	0	43	24	0	32	0	32	0	0
1	5	1	3	858	39	0	38	40	0	0	0	0	0	0	24	39
1	6	1	5	953	0	0	28	0	28	0	0	0	34	0	0	0
1	7	2	8	988	32	0	0	35	43	0	0	40	0	0	0	35
1	8	2	2	1031	40	0	0	38	41	0	0	0	0	0	0	0
1	9	3	2	875	0	0	46	44	39	0	0	0	0	23	0	0
1	10	3	5	1041	0	34	34	30	39	40	0	0	0	39	0	42
2	1	1	3	997	0	0	30	0	35	0	27	0	0	0	41	40
2	2	1	2	1057	30	0	0	0	0	0	0	0	0	28	0	0
2	3	1	3	914	42	0	0	40	27	0	0	0	0	23	0	0
2	4	2	7	925	0	32	0	0	32	0	0	0	34	32	42	0
2	5	1	3	954	0	33	38	0	40	0	0	24	31	0	0	0
2	6	1	8	889	24	32	0	0	43	0	0	0	0	30	35	0
2	7	1	4	873	0	27	0	0	41	0	0	0	28	0	0	0

P	J	W	RR	DD	$M_{1,11}$	$M_{1,12}$	$M_{1,13}$	$M_{1,14}$	$M_{1,15}$	$M_{1,16}$	$M_{1,17}$
					69	76	83	87	91	99	105
Runtime of Jobs											
1	1	1	2	1049	0	0	0	25	30	28	41
1	2	1	6	930	0	42	0	41	39	39	43
1	3	1	7	1022	0	42	0	30	44	41	31
1	4	3	6	940	0	0	36	0	34	29	36
1	5	1	3	858	0	0	33	34	41	29	45
1	6	1	5	953	0	0	28	41	34	33	46
1	7	2	8	988	0	28	33	39	0	41	32
1	8	2	2	1031	36	38	0	41	0	45	28
1	9	3	2	875	44	32	28	27	37	29	32
1	10	3	5	1041	0	0	0	0	45	41	31
2	1	1	3	997	32	29	29	31	31	33	30
2	2	1	2	1057	0	0	0	0	45	28	35
2	3	1	3	914	0	0	0	0	44	45	39
2	4	2	7	925	0	28	0	28	32	38	32
2	5	1	3	954	0	27	37	37	0	46	32
2	6	1	8	889	0	0	0	33	0	31	44
2	7	1	4	873	0	0	30	24	0	37	33

Table E.1 Experimental results for small problem structure

9 Jobs, 17 Stages , 19 Machines, Block 1													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	1845	384	.054	384	.068	295	.065	384	.053	384	.062	384	.068
IS2	1745	487	.090	354	.088	384	.087	384	.088	384	.076	558	.076
IS3	2003	384	.044	412	.048	375	.037	375	.054	384	.058	843	.038
IS4	1355	279	.036	258	.042	251	.041	279	.036	279	.041	689	.041
IS5	1148	384	.047	384	.047	371	.031	380	.055	457	.054	709	.038
9 Jobs, 17 Stages , 19 Machines, Block 2													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	553	64	.025	64	.032	64	.031	64	.029	64	.045	64	.038
IS2	601	64	.041	64	.047	64	.046	64	.048	64	.034	64	.045
IS3	889	77	.079	77	.088	77	.088	73	.069	77	.078	77	.078
IS4	493	64	.038	64	.036	64	.036	64	.038	64	.046	64	.032
IS5	493	64	.071	64	.088	64	.088	64	.075	64	.076	64	.076
9 Jobs, 17 Stages , 19 Machines, Block 3													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	1489	548	.050	596	.034	737	.035	854	.057	605	.045	1184	.043
IS2	1575	548	.076	601	.075	554	.077	645	.070	906	.079	1016	.071
IS3	1602	548	.056	552	.085	596	.078	717	.066	810	.067	823	.068
IS4	1492	596	.074	737	.067	440	.082	803	.082	803	.064	950	.086
IS5	1483	548	.058	596	.086	440	.078	854	.061	605	.066	1184	.066
9 Jobs, 17 Stages , 19 Machines, Block 4													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	186	0	.031	0	.041	0	.046	0	.034	0	.062	0	.062
IS2	202	0	.054	0	.039	0	.033	0	.050	0	.062	0	.078.
IS3	219	0	.056	0	.071	0	.065	0	.035	0	.062	0	.078
IS4	151	0	.044	0	.042	0	.040	0	.037	0	.060	0	.080
IS5	151	0	.041	0	.061	0	.068	0	.040	0	.059	0	.044
9 Jobs, 17 Stages , 19 Machines, Block 5													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	3988	2945	.035	2241	.031	3026	.031	3701	.035	3701	.036	3290	.044
IS2	3512	2564	.059	2545	.066	2865	.065	2456	.055	2403	.064	2314	.056
IS3	3536	2778	.045	2777	.058	2817	.054	2978	.045	2943	.044	2464	.057
IS4	3168	2943	.030	2943	.036	2863	.043	3026	.053	2943	.068	2737	.042
IS5	3465	2737	.044	2737	.054	2984	.058	2943	.045	2943	.039	2737	.059

12 Jobs, 17 Stages , 19 Machines, Block 1

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	576	201	.093	166	.046	186	.046	321	.094	317	.042	414	.043
IS2	654	345	.045	345	.046	186	.046	345	.050	317	.056	301	.054
IS3	603	287	.087	147	.089	145	.098	416	.095	354	.089	443	.096
IS4	554	196	.033	121	.053	126	.040	203	.037	215	.058	369	.046
IS5	554	297	.085	247	.098	300	.098	335	.095	279	.098	374	.098

12 Jobs, 17 Stages , 19 Machines, Block 2

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	756	356	.048	543	.037	598	.031	356	.047	469	.038	664	.035
IS2	884	198	.065	611	.058	562	.082	291	.070	436	.061	676	.064
IS3	818	471	.045	546	.050	523	.054	543	.047	354	.046	531	.086
IS4	901	512	.061	546	.075	602	.068	543	.055	279	.078	223	.062
IS5	830	385	.040	543	.056	332	.051	350	.047	249	.046	276	.084

12 Jobs, 17 Stages , 19 Machines, Block 3

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	1190	329	.056	319	.062	298	.065	578	.066	578	.063	654	.065
IS2	983	345	.071	465	.050	223	.054	321	.070	421	.087	689	.059
IS3	875	300	.097	441	.097	410	.090	300	.085	450	.100	637	.098
IS4	759	367	.066	319	.047	356	.096	383	.067	389	.047	577	.088
IS5	552	304	.097	337	.096	332	.090	397	.088	335	.100	512	.098

12 Jobs, 17 Stages , 19 Machines, Block 4

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	187	0	.046	0	.078	0	.075	0	.056	0	.062	0	.064
IS2	268	0	.070	0	.099	0	.064	0	.064	0	.126	0	.095
IS3	93	0	.034	0	.098	0	.070	0	.032	0	.087	0	.069
IS4	56	0	.051	0	.078	0	.020	0	.050	0	.020	0	.030
IS5	70	0	.025	0	.074	0	.092	0	.075	0	.081	0	.131

12 Jobs, 17 Stages , 19 Machines, Block 5

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	3698	2086	.049	2086	.046	1649	.046	2215	.056	2215	.042	2906	.062
IS2	3559	1334	.047	1856	.054	1316	.040	1648	.056	2680	.051	2606	.060
IS3	3475	762	.083	1978	.059	741	.065	1021	.034	1943	.047	2359	.045
IS4	3876	1649	.045	1649	.048	1018	.052	2640	.081	2640	.064	3067	.040
IS5	3338	1925	.080	2295	.058	1649	.060	2412	.047	2412	.048	2309	.046

17 Jobs, 17 Stages , 19 Machines, Block 1

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	1683	1482	.093	1250	.102	1275	.109	1124	.094	1035	.067	978	.114
IS2	1456	1201	.153	1183	.103	975	.126	1292	.149	1257	.089	924	.123
IS3	951	681	.192	652	.145	593	.158	644	.147	629	.197	788	.141
IS4	1029	800	.093	756	.100	593	.109	597	.088	597	.080	788	.104
IS5	1212	687	.199	656	.222	644	.216	644	.169	543	.168	701	.165

17 Jobs, 17 Stages , 19 Machines, Block 2

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	962	618	.094	618	.103	618	.109	618	.087	618	.062	618	.105
IS2	947	618	.170	618	.199	618	.188	618	.166	618	.188	618	.199
IS3	1433	618	.127	612	.142	618	.147	618	.122	618	.153	618	.148
IS4	676	612	.083	612	.065	612	.079	612	.082	612	.084	612	.092
IS5	951	612	.127	612	.142	612	.147	612	.122	612	.155	612	.145

17 Jobs, 17 Stages , 19 Machines, Block 3

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	1571	365	.156	365	.271	198	.296	709	.152	487	.174	890	.224
IS2	1601	659	.238	575	.214	254	.211	547	.225	422	.213	953	.215
IS3	2075	452	.250	248	.270	442	.280	753	.230	722	.240	793	.255
IS4	2308	232	.166	232	.224	220	.213	982	.166	1000	.220	652	.212
IS5	2142	92	.250	92	.270	91	.280	615	.230	615	.240	532	.260

17 Jobs, 17 Stages , 19 Machines, Block 4

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	155	0	.093	0	.109	0	.124	0	.093	0	.249	0	.124
IS2	264	0	.114	0	.395	0	.355	0	.198	0	.272	0	.156
IS3	188	0	.117	0	.311	0	.318	0	.215	0	.225	0	.254
IS4	96	0	.109	0	.062	0	.109	0	.066	0	.124	0	.062
IS5	121	0	.167	0	.188	0	.191	0	.136	0	.248	0	.233

17 Jobs, 17 Stages , 19 Machines, Block 5

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	5494	1974	.171	2244	.218	2779	.234	3529	.176	3091	.217	2924	.224
IS2	4578	3564	.228	2681	.263	1987	.231	3214	.227	2457	.230	3655	.225
IS3	4666	3874	.250	2475	.235	2654	.260	3475	.300	2645	.261	3661	.302
IS4	4077	2646	.140	2646	.211	2607	.218	3752	.151	2705	.200	3914	.209
IS5	4270	2646	.256	2607	.247	2837	.262	3139	.306	2382	.341	3161	.300

25 Jobs, 17 Stages , 19 Machines, Block 1

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	440	74	.30	74	.47	74	.55	74	.27	74	.39	74	.37
IS2	705	53	.23	53	.31	74	.41	161	.20	161	.38	133	.41
IS3	667	134	.35	134	.67	89	.78	105	.29	74	.30	111	.41
IS4	517	71	.27	71	.74	74	.38	74	.34	74	.45	74	.56
IS5	414	84	.20	74	.43	74	.43	74	.32	74	.60	74	.60

25 Jobs, 17 Stages , 19 Machines, Block 2

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	488	321	.54	321	.53	321	.68	321	.46	321	.75	321	.62
IS2	496	321	.43	321	.62	321	.88	321	.58	321	.75	321	.63
IS3	1168	321	.28	321	.79	321	.94	321	.40	321	.82	321	.59
IS4	470	321	.47	321	.72	321	.88	321	.62	321	.85	321	.80
IS5	391	321	.31	321	.72	321	.73	321	.37	321	.63	321	.61

25 Jobs, 17 Stages , 19 Machines, Block 3

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	776	680	.30	514	.47	559	.55	669	.27	581	.39	495	.37
IS2	793	680	.45	612	.54	559	.67	658	.27	543	.61	652	.61
IS3	948	576	.43	548	.62	566	.63	579	.20	543	.84	547	.77
IS4	735	576	.26	548	.48	549	.75	579	.59	543	.58	543	.76
IS5	714	680	.40	543	.92	543	.70	543	.42	543	.82	543	.84

25 Jobs, 17 Stages , 19 Machines, Block 4

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	143	41	.44	27	.60	24	.71	41	.41	27	.98	24	.82
IS2	173	41	.42	41	.70	24	.80	41	.55	41	.98	24	.75
IS3	556	13	.59	13	.77	16	.68	16	.36	13	.76	15	.64
IS4	195	48	.64	29	.82	41	.76	59	.49	38	.80	42	.64
IS5	210	48	.31	29	.75	43	.48	49	.56	38	.81	47	.75

25 Jobs, 17 Stages , 19 Machines, Block 5

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	6317	3409	.48	3279	.75	4554	1.57	5247	.17	4497	.43	4800	.72
IS2	4593	3547	.31	2978	.71	4009	.98	3512	.32	3510	.88	3854	.90
IS3	5478	4517	.26	4517	.52	4918	.98	4580	.46	3936	.88	5162	.95
IS4	5662	4987	.48	3681	.60	3785	.83	4257	.25	4262	.80	5125	.58
IS5	4531	4325	.59	3674	.81	4128	.70	3451	.19	3451	.64	3587	.47

35 Jobs, 17 Stages , 19 Machines, Block 1

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	108	38	2.37	38	5.37	38	4.71	38	3.08	38	4.80	38	5.69
IS2	108	38	1.76	38	5.39	38	4.10	38	4.11	38	5.05	38	6.57
IS3	277	76	2.04	76	4.53	76	4.04	76	3.01	76	5.78	76	4.71
IS4	102	38	2.14	38	2.61	38	4.09	38	3.87	38	6.83	38	6.43
IS5	83	38	2.00	38	5.19	38	3.82	38	3.86	38	5.83	38	5.09

35 Jobs, 17 Stages , 19 Machines, Block 2

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	1681	411	3.75	411	5.62	411	6.05	411	4.42	411	5.15	411	5.00
IS2	1342	420	2.71	407	6.39	443	4.27	483	4.39	407	5.86	404	6.32
IS3	1256	469	4.80	469	5.25	410	4.80	536	3.23	479	6.07	408	7.84
IS4	659	389	2.13	385	5.22	381	2.68	389	3.25	386	7.42	381	5.29
IS5	1450	417	3.11	354	4.37	358	4.56	417	3.26	412	6.46	381	5.08

35 Jobs, 17 Stages , 19 Machines, Block 3

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	2415	221	2.11	360	7.99	117	6.64	221	1.33	266	4.09	360	5.74
IS2	2580	203	2.62	196	6.05	193	5.52	196	3.22	110	4.61	196	4.50
IS3	2412	240	3.69	151	4.27	153	6.63	240	3.25	151	7.84	151	5.59
IS4	1585	248	2.52	151	6.80	153	6.39	248	3.23	151	5.90	151	4.77
IS5	1585	248	2.23	151	6.45	153	4.18	248	2.00	151	5.90	151	4.00

35 Jobs, 17 Stages , 19 Machines, Block 4

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	79	0	2.87	0	4.43	0	6.82	0	2.21	0	4.68	0	6.76
IS2	87	0	2.67	0	5.83	0	3.96	0	2.52	0	4.61	0	3.65
IS3	63	0	3.44	0	5.41	0	4.80	0	3.70	0	7.74	0	4.97
IS4	59	0	3.12	0	4.63	0	5.81	0	2.74	0	3.59	0	3.59
IS5	59	0	3.45	0	4.02	0	5.07	0	3.11	0	4.39	0	4.39

35 Jobs, 17 Stages , 19 Machines, Block 5

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	7671	2721	2.8	2637	4.90	4222	4.61	2701	2.22	2088	3.98	4764	2.74
IS2	9595	2725	2.26	2870	4.10	6363	5.17	2597	2.93	2112	5.15	5124	4.16
IS3	8644	2968	3.15	2870	5.15	6620	5.19	3896	3.05	2988	5.00	3626	5.00
IS4	8562	2764	2.75	2632	5.15	3793	4.44	2755	2.85	2755	4.28	2707	5.21
IS5	8774	2527	3.00	2801	6.21	2684	5.65	2618	2.97	2600	3.95	2618	6.03

45 Jobs, 17 Stages , 19 Machines, Block 1

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	2187	1410	5.6	1410	7.04	1410	6.32	1410	4.78	1410	6.97	1410	7.85
IS2	3245	1421	4.18	1405	8.25	1404	7.19	1419	3.99	1405	8.33	1404	8.33
IS3	1077	778	5.14	756	9.87	997	8.98	778	5.46	756	9.54	997	8.54
IS4	1204	864	6.00	864	8.12	921	8.02	950	4.74	864	9.54	904	7.77
IS5	1260	1013	5.84	732	10.02	745	9.97	941	5.19	941	8.79	885	8.98

45 Jobs, 17 Stages , 19 Machines, Block 2

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	1161	420	3.98	420	7.71	420	8.30	420	3.98	420	7.71	420	8.30
IS2	1257	417	4.64	402	8.86	400	7.12	417	4.45	406	6.44	404	6.44
IS3	1576	469	5.16	469	7.25	419	7.25	503	4.36	479	7.71	410	6.71
IS4	659	384	4.78	384	6.38	383	8.03	386	5.19	385	8.23	401	7.91
IS5	1218	647	4.55	647	6.02	845	6.95	647	4.55	647	6.02	845	6.95

45 Jobs, 17 Stages , 19 Machines, Block 3

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	4671	2252	6.67	2175	9.19	2693	9.78	2252	6.21	2175	8.64	2693	9.26
IS2	4887	2275	5.47	2175	8.12	2693	8.12	2275	4.96	2175	7.17	2693	7.02
IS3	5595	2968	4.85	2870	6.76	2663	6.76	2968	5.12	2870	8.23	2663	8.16
IS4	5628	2676	5.02	2756	7.55	2654	6.57	2676	5.02	2756	7.55	2654	6.57
IS5	5272	2600	4.12	2600	6.31	2754	6.31	2600	4.12	2600	6.31	2754	6.31

45 Jobs, 17 Stages , 19 Machines, Block 4

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	117	0	3.24	0	6.23	0	5.87	0	3.24	0	6.23	0	5.87
IS2	203	0	4.18	0	7.12	0	6.57	0	4.18	0	7.12	0	6.57
IS3	218	0	4.06	0	7.47	0	8.15	0	5.12	0	6.85	0	7.19
IS4	93	0	5.45	0	7.68	0	7.68	0	5.45	0	7.68	0	7.68
IS5	93	0	3.97	0	6.29	0	6.29	0	4.52	0	8.25	0	8.36

45 Jobs, 17 Stages , 19 Machines, Block 5

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	4478	3620	5.23	3329	8.96	3620	7.69	3112	5.01	3112	8.65	3805	7.71
IS2	4585	3529	4.78	3529	8.24	3594	8.56	3259	5.62	3125	9.12	3592	8.12
IS3	8296	3543	6.02	3515	9.68	3242	9.68	3309	6.78	2943	10.21	3423	9.43
IS4	3825	3572	5.64	3223	10.23	3110	9.51	3276	4.25	3006	8.75	3257	9.01
IS5	3825	3572	5.12	3223	9.85	3110	8.79	3276	5.18	3006	7.28	3257	8.62

55 Jobs, 17 Stages , 19 Machines, Block 1

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	450	123	9.12	123	13.45	231	14.73	143	9.12	143	13.45	221	14.73
IS2	651	105	10.45	101	14.50	220	15.07	112	11.77	105	15.50	264	13.96
IS3	752	250	9.48	210	13.85	342	12.82	234	10.78	143	14.18	144	15.81
IS4	347	112	11.70	98	15.09	178	14.10	117	8.89	98	13.20	124	15.20
IS5	347	112	12.15	98	16.78	178	17.61	117	10.62	98	14.44	124	13.26

55 Jobs, 17 Stages , 19 Machines, Block 2

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	778	351	8.16	251	14.12	98	13.25	351	8.16	251	12.04	98	13.83
IS2	576	98	9.12	85	15.18	98	14.32	98	9.58	85	13.81	98	14.10
IS3	734	130	8.19	84	13.24	112	14.36	84	10.45	84	15.45	83	15.45
IS4	643	197	10.65	125	15.97	121	15.97	197	8.64	129	14.17	121	15.48
IS5	643	197	11.23	125	16.61	121	16.11	197	10.72	129	14.75	121	15.50

55 Jobs, 17 Stages , 19 Machines, Block 3

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	3587	2513	8.15	1709	13.45	1395	14.65	2513	8.14	1709	12.82	1395	15.75
IS2	4904	1772	9.23	1778	14.45	1778	14.82	1772	9.40	1778	13.58	1778	13.58
IS3	6752	1932	10.14	1393	13.68	1932	13.72	1932	8.65	1393	13.27	1932	13.27
IS4	5899	1945	8.45	1419	14.93	1800	15.30	1945	10.17	1419	16.55	1800	14.59
IS5	5899	1945	9.38	1419	15.84	1800	14.68	1945	11.45	1419	17.18	1800	16.03

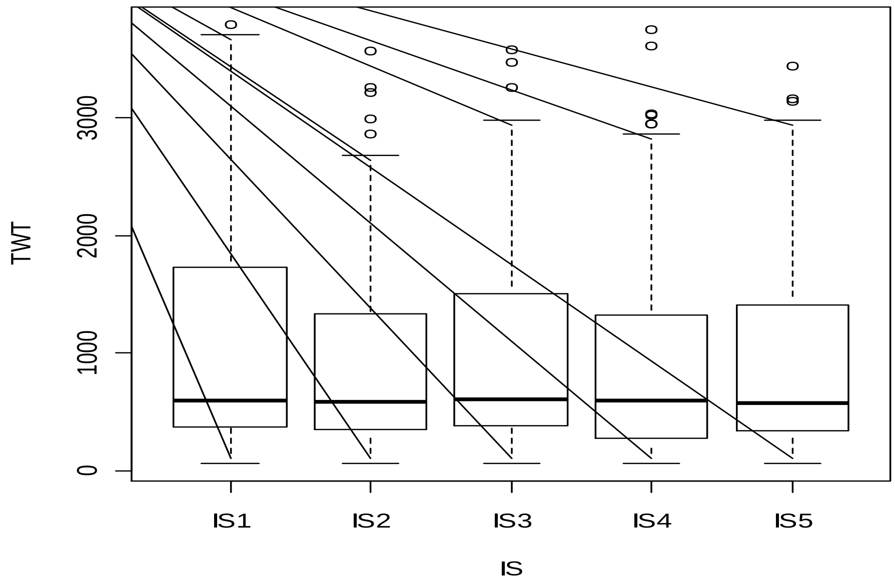
55 Jobs, 17 Stages , 19 Machines, Block 4

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	210	0	9.12	0	13.54	0	14.56	0	7.17	0	12.75	0	13.24
IS2	179	0	10.16	0	13.72	0	12.62	0	10.10	0	14.13	0	13.54
IS3	98	0	8.15	0	11.76	0	14.18	0	10.45	0	12.68	0	14.87
IS4	67	0	10.73	0	12.82	0	13.19	0	9.73	0	13.82	0	16.13
IS5	154	0	9.56	0	13.97	0	12.85	0	8.22	0	11.91	0	11.47

55 Jobs, 17 Stages , 19 Machines, Block 5

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	9495	4412	9.42	4407	13.85	5759	12.16	4214	10.12	4214	15.32	5662	13.26
IS2	8516	4468	10.11	4386	14.25	5212	14.65	3741	11.47	3642	14.69	4713	13.26
IS3	7521	4435	9.17	3620	13.46	4622	17.18	4872	9.83	4800	16.71	6471	12.82
IS4	6863	4393	11.73	4393	14.89	3541	11.29	3842	10.72	3125	16.29	4275	14.26
IS5	8132	5463	10.46	4545	15.41	4587	13.81	7821	11.03	5740	15.01	4834	17.23

Appendix F. ANALYSIS OF EXPERIMENTAL RESULTS (TOTAL WEIGHTED TARDINESS)



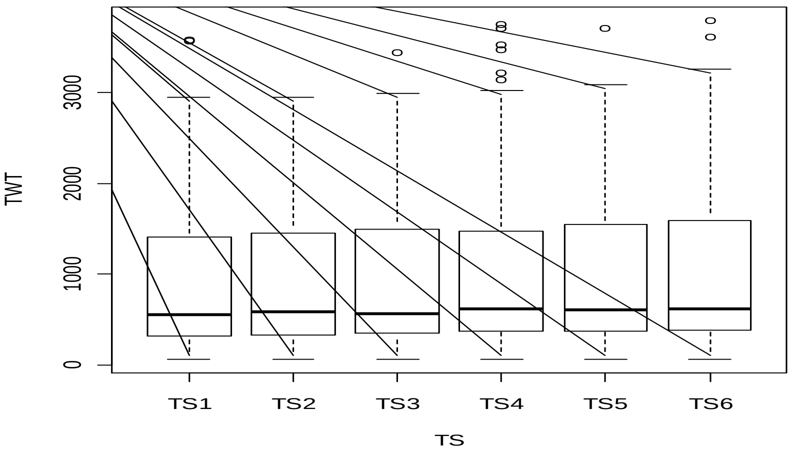
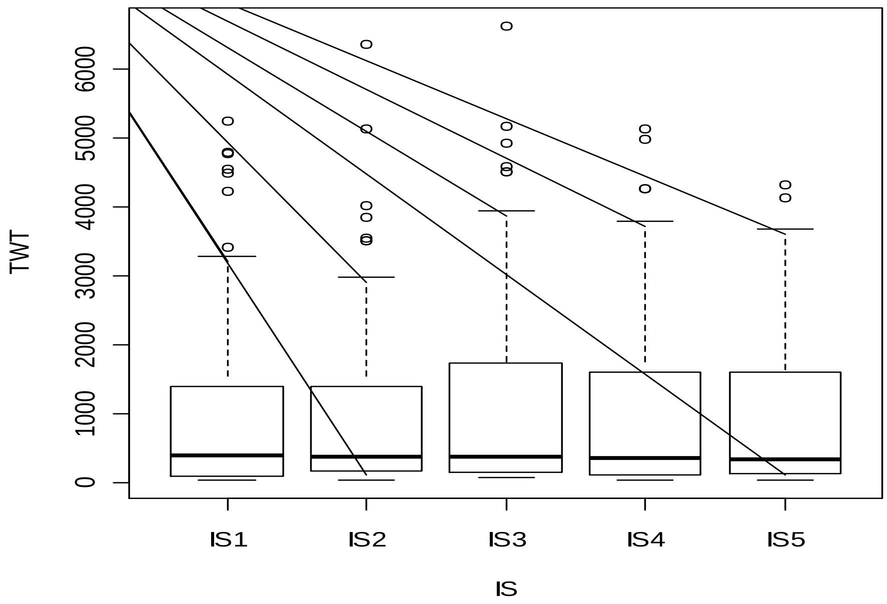


Figure F.1 Box Plots of total weighted tardiness between (a) levels of IS; (b) levels of TS for small problem structures



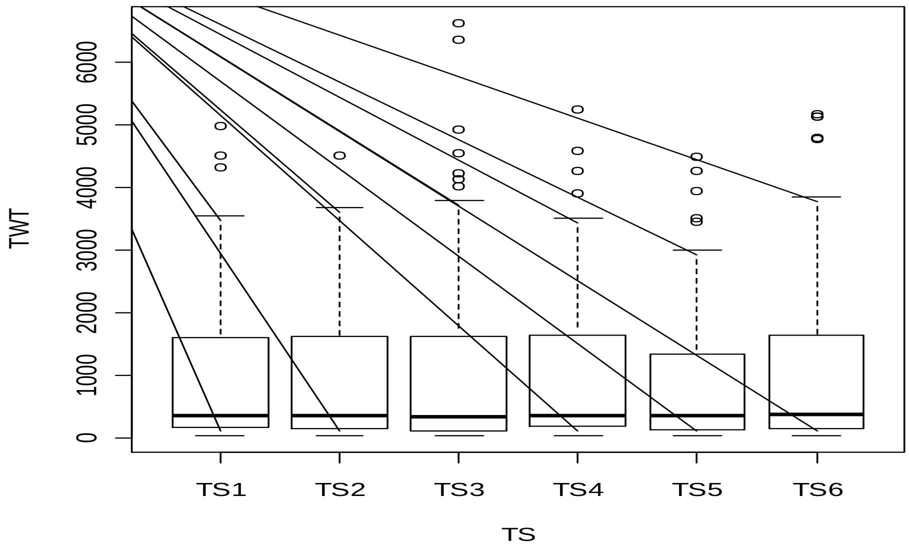
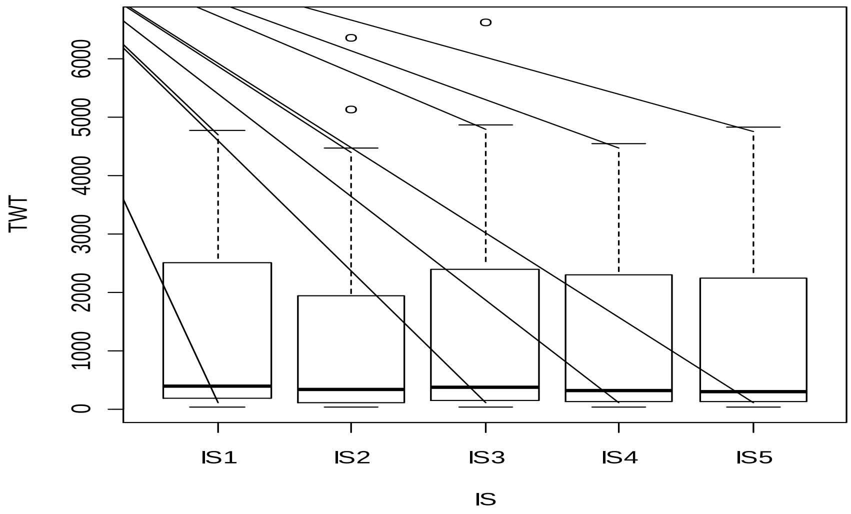


Figure F.2 Box Plots of total weighted tardiness between (a) levels of IS; (b) levels of TS for medium problem structures



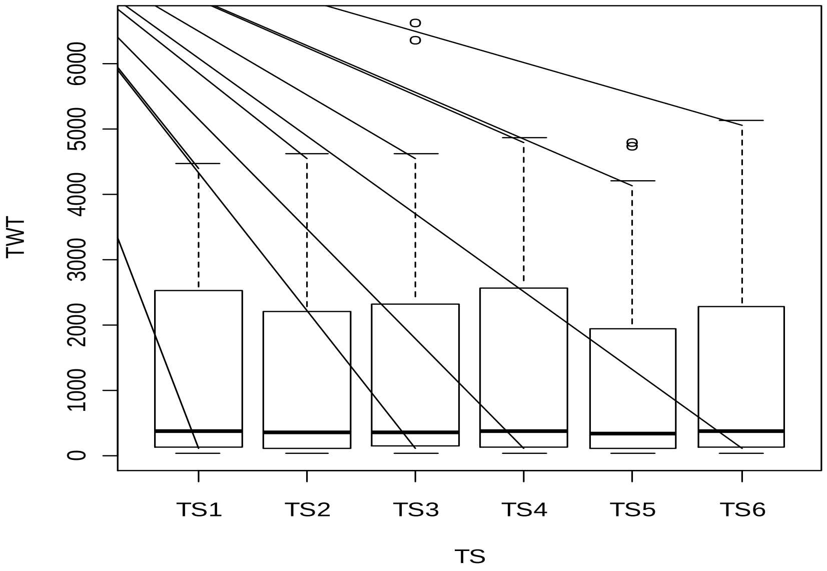


Figure F.3 Box Plots of total weighted tardiness between (a) levels of IS; (b) levels of TS for large problem structures

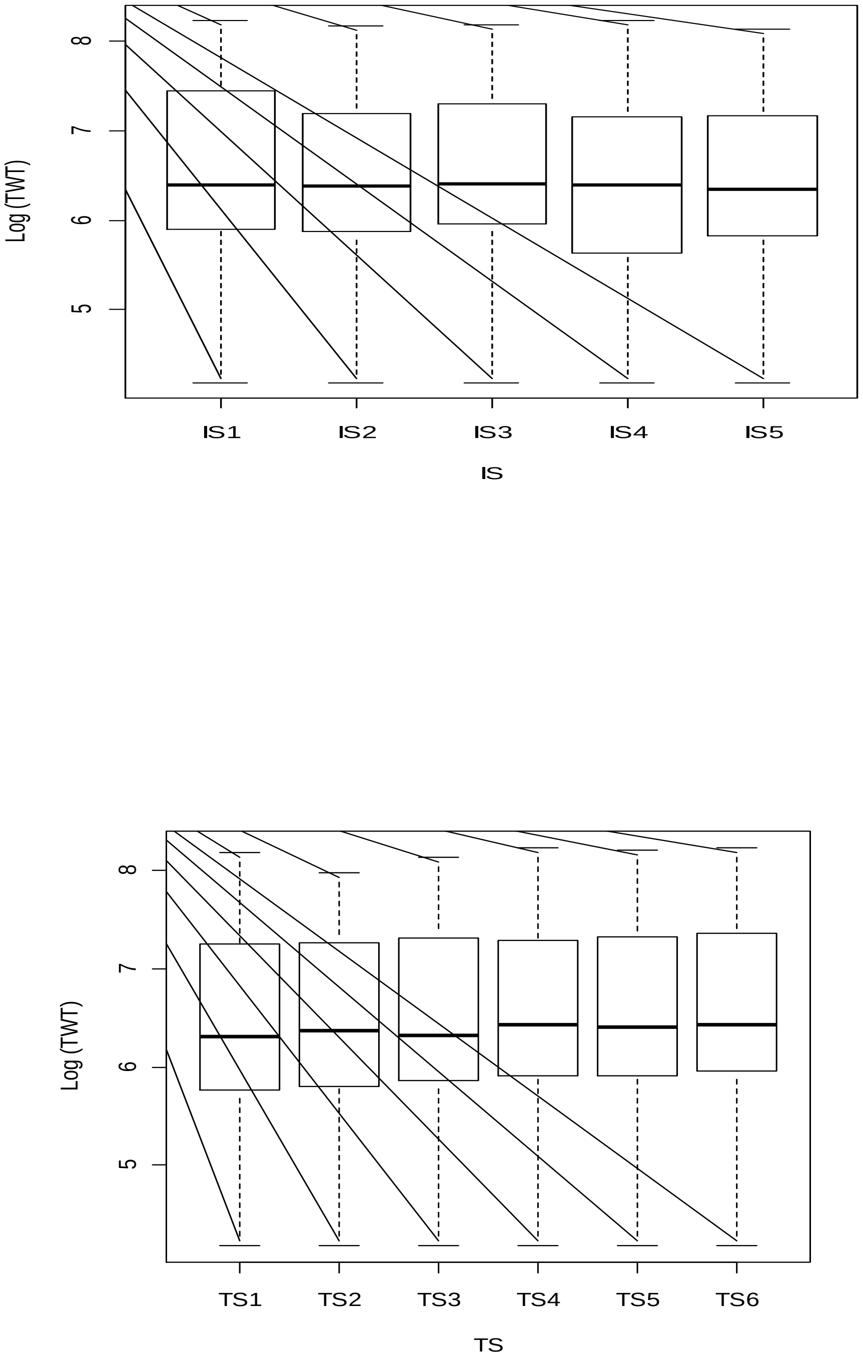


Figure F.4 Box Plots of Log(TWT) between (a) levels of IS; (b) levels of TS for small problem structure

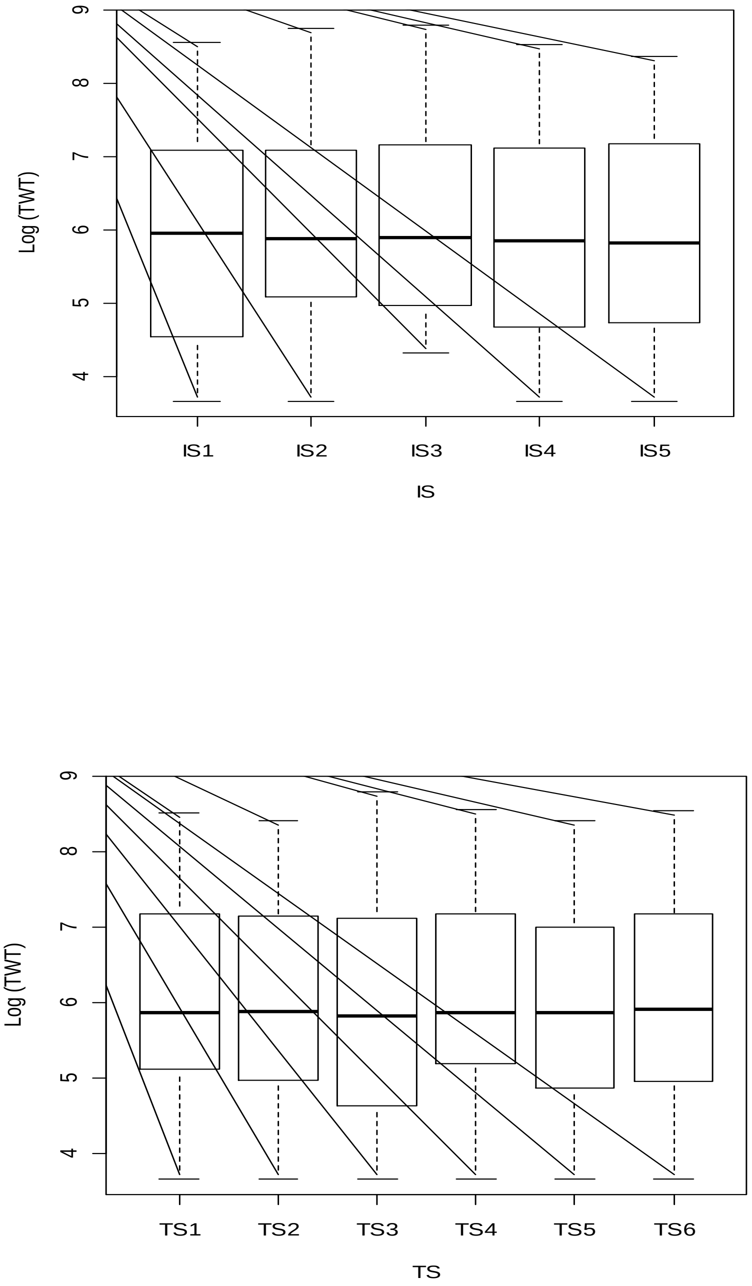
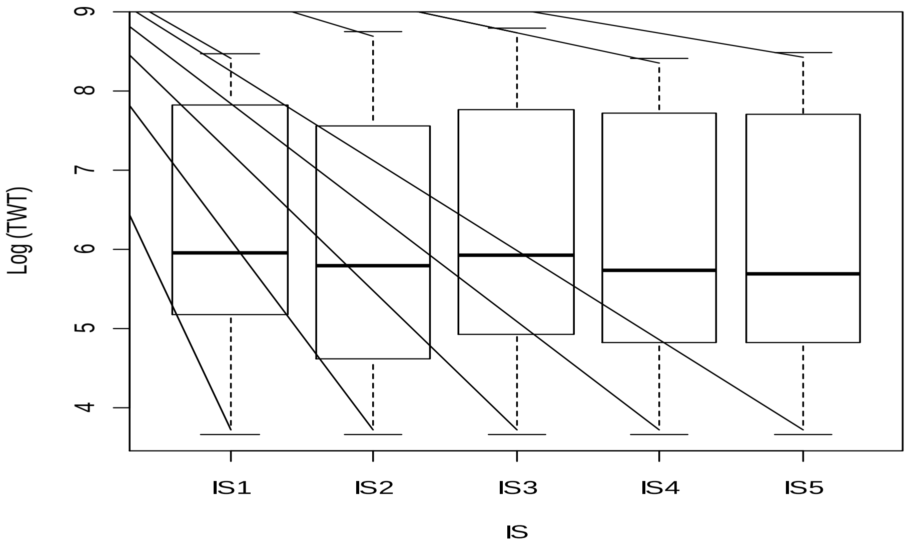


Figure F.5 Box Plots of Log(TWT) between (a) levels of IS; (b) levels of TS for medium problem structure



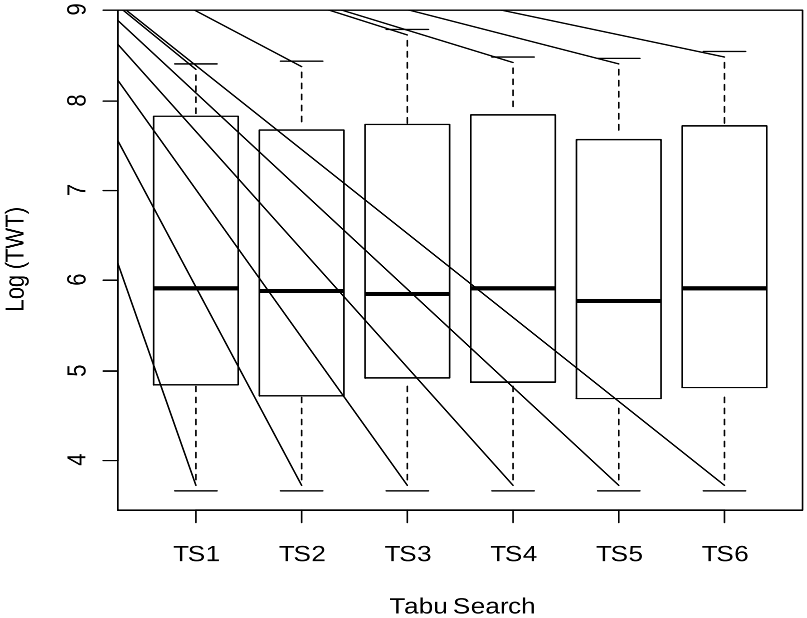
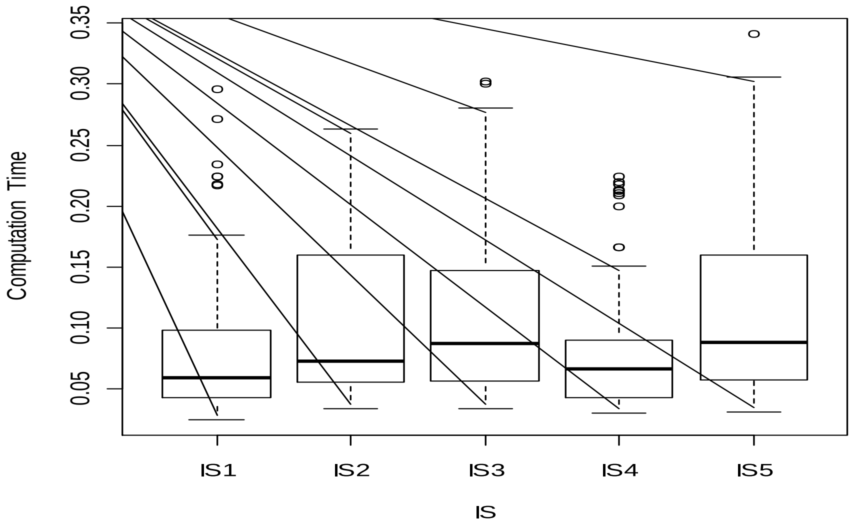


Figure F.6 Box Plots of Log(TWT) between (a) levels of IS; (b) levels of TS for the small problem structure

APPENDIX G. ANALYSIS OF EXPERIMENTAL RESULTS (COMPUTATION TIME)



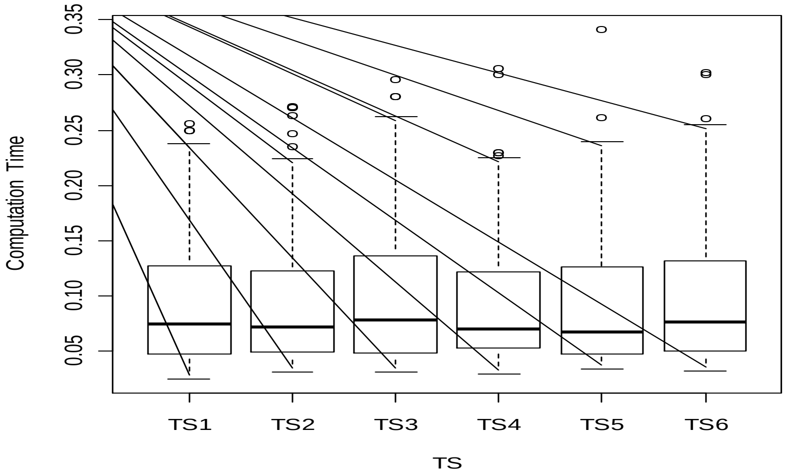
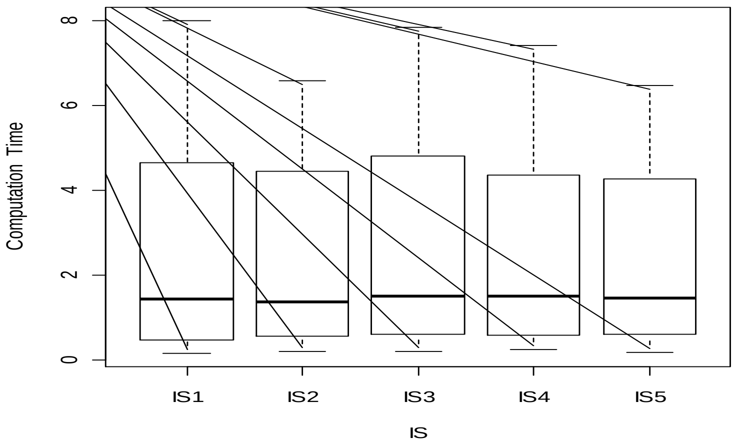


Figure G.1 Box Plots of computation time between (a) levels of IS; (b) levels of TS for small problem structure



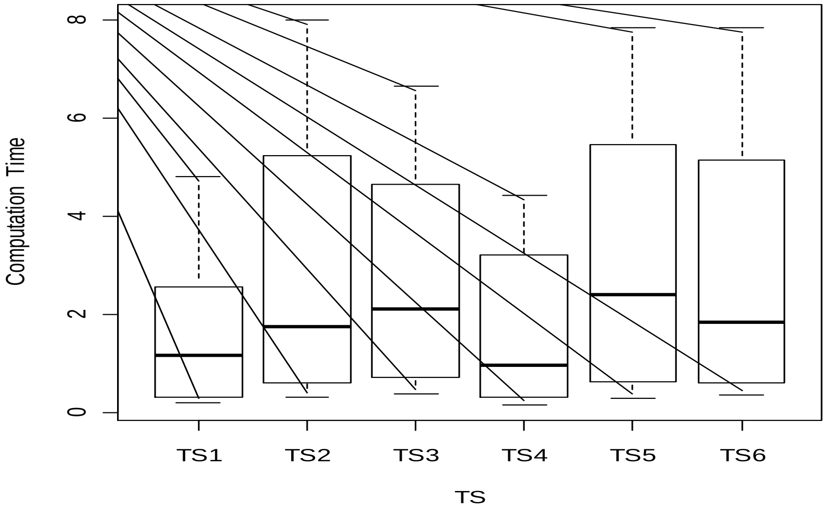
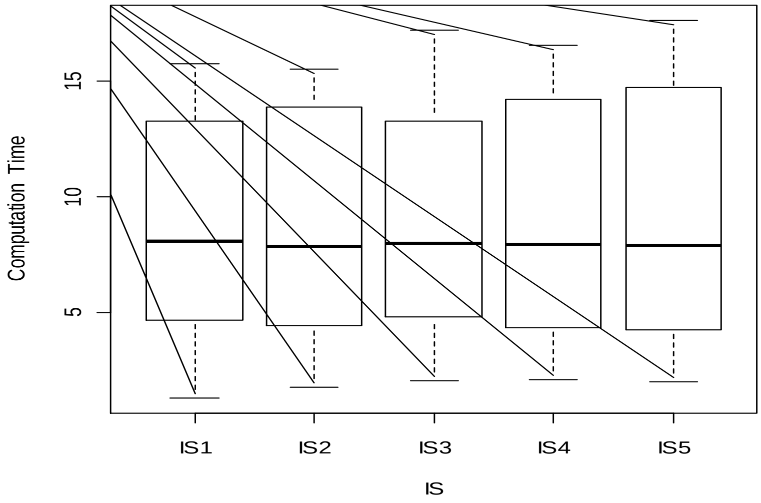


Figure G.2 Box Plots of computation time between (a) levels of IS; (b) levels of TS for medium problem structure



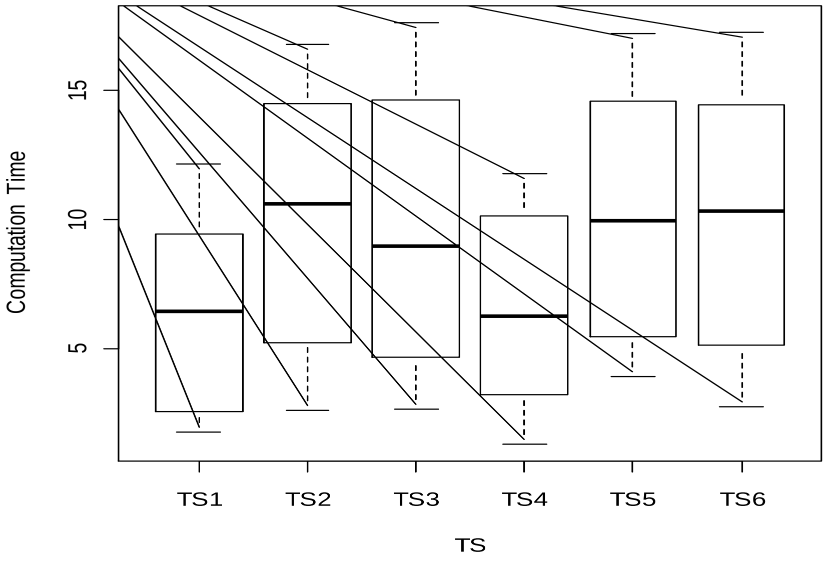
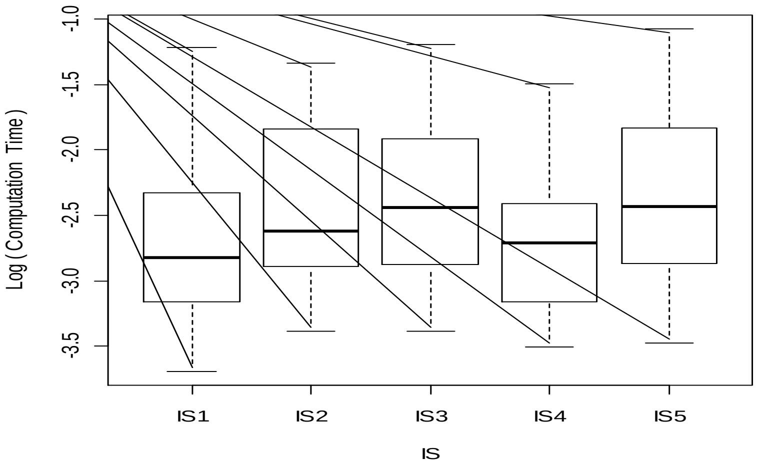


Figure G.3 Box Plots of computation time between (a) levels of IS; (b) levels of TS for large problem structure



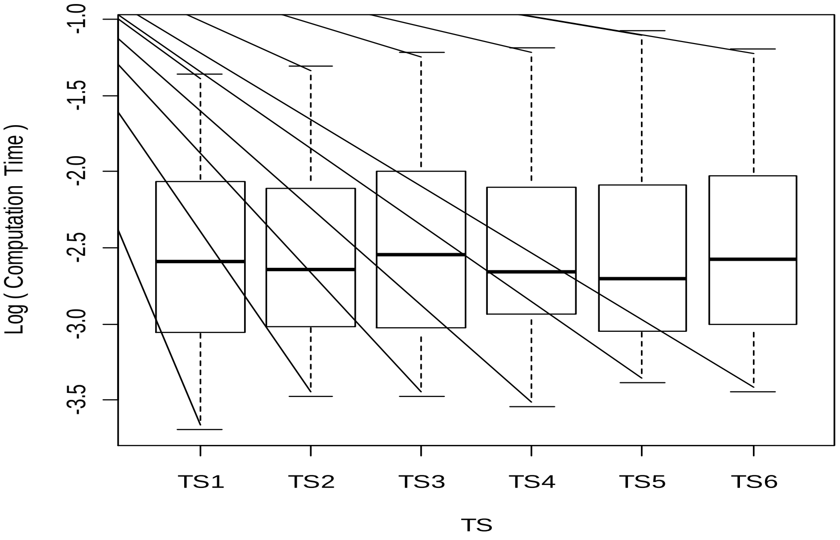
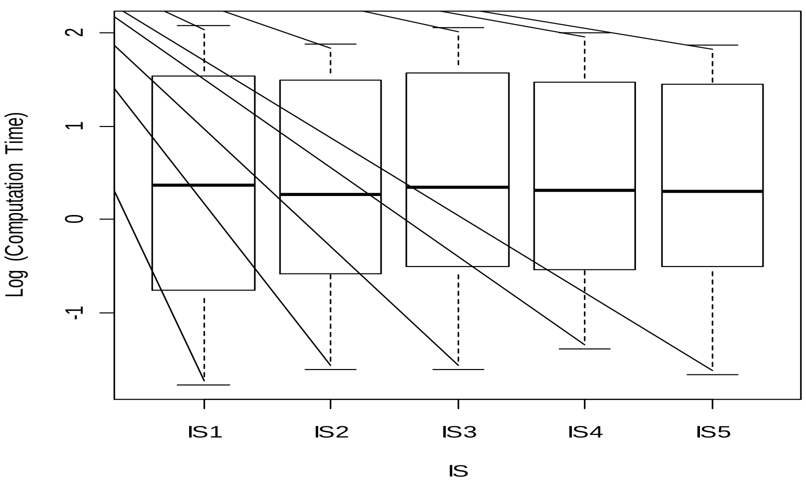


Figure G.4 Box Plots of Log (Computation Time) between (a) levels of IS; (b) levels of TS for small problem structure



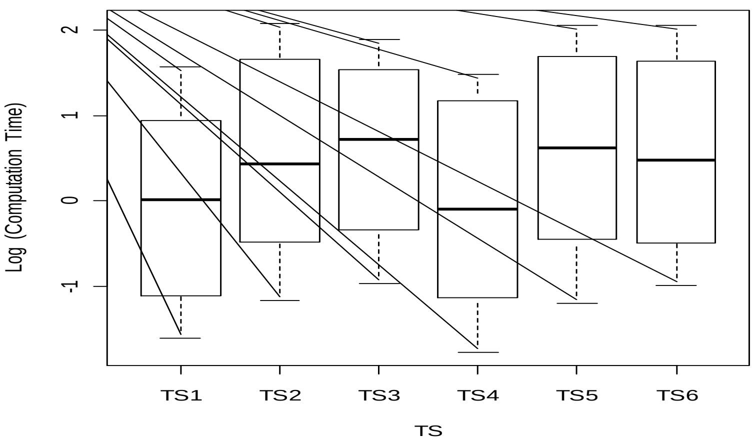


Figure G.5 Box Plots of Log (Computation Time) between (a) levels of IS; (b) levels of TS for medium problem structure