AN ABSTRACT OF THE THESIS OF

Fenny Subur for the degree of Master of Science in Industrial Engineering presented on April 27, 2000. Title: A Methodology for Real-Time Scheduling of Jobs with Splitting on Unrelated Parallel Machines.

Abstract approved: Redacted for Privacy

Rasaratnam Logendran

Unrelated parallel machines are machines that perform the same function but have different capacity or capability. Thus, the processing time of each job would be different on machines of different types. The scheduling environment considered is dynamic in both job release time and machine availability. Additionally, each job considered can have different weight, and due date. Split-jobs are also considered in this research. The number of jobs that needs to be processed in split-modes is pre-determined and not part of the scheduling decision. Additional constraints are imposed on split jobs to ensure that the absolute difference in completion time of the split portions of a job is within a user-specified margin. These constraints are supported by the Just-In-Time manufacturing concept where inventory has to be maintained at a very low or zero level. 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 it belongs to the class of NP-hard problems. Thus, one cannot rely on using an implicit enumeration technique, such as the one based on branch-and-bound, to solve industry-size problems within a reasonable computation time. Therefore, a higher-level search heuristic, based on a concept known as tabu search, is developed to solve the problems. Four different methods based on simple and composite dispatching rules are used to generate the initial solution that is used by tabu-search as a starting point. Six different tabu-search based heuristics are developed by incorporating the different features of tabu search. The heuristics are tested on eight small problems and the quality

of their solutions is compared to their optimal solutions, which are obtained by applying the branch-and-bound technique. The evaluation shows that the tabu-search based heuristics are capable of obtaining solutions of good quality within a much shorter time. The best performer among these heuristics recorded a percentage deviation of only 1.18%.

The performance of the tabu-search based heuristics is compared by conducting a statistical experiment that is based on a split-plot design. Three sizes of problem structures, ranging from 9 jobs to 60 jobs and from 3 machines to 15 machines are used in the experiment. The results of the experiment reveal that in comparison to other initial-generation methods, the composite dispatching rule is capable of obtaining initial solutions that significantly accelerate the tabu-search based heuristic to get to the final solution. The use of long-term memory function is proven to be advantageous in solving all problem structures. The long-term memory based on maximum-frequency strategy is recommended for solving the small problem structure, while the minimum-frequency strategy is preferred for solving medium and large problem structures. With respect to the use of tabu-list size as a parameter, the variable tabu-list size is preferred for solving the smaller problem structure, but the fixed tabu-list size is preferred as the size of the problems grows from small to medium and then large.

A Methodology for Real-Time Scheduling of Jobs with Splitting on Unrelated Parallel Machines

by

Fenny Subur

A THESIS
submitted to
Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented April 27, 2000
Commencement June 2000

Master of Science thesis of Fenny Subur presented on April 27, 2000

APPROVED:

Redacted for Privacy

Major Professor, representing Industrial Engineering

Redacted for Privacy

Chair of Department of Industrial and Manufacturing Engineering

Redacted for Privacy

Dean of 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.

Redacted for Privacy

Fenny Subur, Author

ACKNOWLEDGEMENT

Many people have made significant contributions to this thesis. They have given continuous support and encouragement that have been a tremendous help to me. I appreciate all of them very much.

My sincere gratitude goes to Dr. Rasaratnam Logendran, my Major Professor, from whom the idea of the research originates. He has continuously advised me throughout the pursuit of my research. His knowledge in real-time scheduling and his commitment to the academic and research field have greatly inspired me.

I would like to express my gratitude to the faculty of the Industrial and Manufacturing Engineering (IME) Department for providing me with a teaching assistantship during part of my graduate study at Oregon State University. I am grateful to the staff of IME, especially Budiyoso Kurniawan, who helped me in setting up the computers for the research purpose.

I would like to thank my committee members: Dr. Sabah Randhawa, my Minor Professor, Dr. John Shea, my committee member, and Dr. Toshimi Minoura, my graduate council representative, for taking the time to evaluate my thesis.

I would also like to thank the Statistics Counseling Service from the Department of Statistics for their assistance in part of my data analysis.

Finally, I would like to express my deepest gratefulness to my parents, my sisters, and all my friends for their assistance, prayers, support and encouragement. Kezia Emerald, Dewi Intan, and Rennya Liyanti for the friendship, understanding and patience. Yunan Karim for the information he shared in the research and experimentation. Tuanjai Somboonwiwat, Patcharaporn Neammanee, and Jirachai Buddhakulsomsiri for their help in statistical analysis.

TABLE OF CONTENTS

Page

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF TABLES

LIST OF TABLES (Continued)

LIST OF APPENDICES

LIST OF APPENDIX FIGURES

LIST OF APPENDIX TABLES

LIST OF APPENDIX TABLES (Continued)

This thesis is dedicated to

JESUS CHRIST

A METHODOLOGY FOR REAL-TIME SCHEDULING OF JOBS WITH SPLITTING ON UNRELATED PARALLEL MACHINES

1. INTRODUCTION

A scheduling problem consists of two components: the machine configuration and the job characteristics. Generally, machine configuration is categorized into single machine, parallel machines, flow shop, and job shop settings. Cheng and Sin (1990) listed five characteristics of a job: job processing time, due-date requirement, preemptive sequencing, precedence constraints, and job release time. The first two job characteristics are self-explanatory. The third characteristic allows an operation of a job to be interrupted and the machine is taken over by another job that is considered to be more urgent. The precedence constraints determine the order in which the jobs have to be processed. In a scheduling problem, if the jobs are released at different times, the condition is called dynamic. Otherwise, it is a static condition. Similar to jobs, machines may be released at different times, which imply dynamic machine availability. Thus, the static and dynamic terms are also applied to machine availability time.

There are three types of parallel machines systems: identical, uniform and unrelated parallel machines. The difference between these parallel machines systems is characterized by a job's processing time among the machines in parallel. In identical parallel machines system, the processing time of a job is the same on all machines in parallel. In uniform parallel machines, each machine has a unique speed factor that determines jobs' processing time. Thus, the processing time of a job on each machine varies by the speed factor of the machine. In unrelated parallel machines, the processing time of a job varies arbitrarily between the machines. Identical parallel machines can be viewed as a reduced version of uniform parallel machines, which in turn is the reduced version of unrelated parallel machines. Unrelated parallel machines can be regarded as machines that perform the same function but have different capability or capacity. Unrelated parallel machines are very common in the industry. A company may invest in similar machines that have different capability, taking into consideration the capital cost, operation cost, and variation in production demand. Therefore, scheduling tasks on unrelated parallel machines is an activity that is very much a part of industry scheduling. Many research efforts have been performed on unrelated parallel machines scheduling. Among the reported studies are Davis and Jaffe, 1979, Lenstra et al., 1987, Hariri and Potts, 1991, Suresh and Chauduri, 1994, Glass et al, 1994, Piersma and Van Dijk, 1996, Suresh and Chauduri, 1996a and 1996b. These studies will be reviewed briefly in the next chapter.

Jobs that compete for limited resources, i.e. a set of unrelated parallel machines, may have different levels of priority and due date. Factors that contribute to setting due date of jobs, to mention a few, are customer requirements, resources' capacity, and shop congestion level. A job with tight due date, high priority, and/or high workload, may need to be split and processed on two machines in parallel. The need for splitting jobs typically appears in an operation that imposes large workload on a machine and requires the entire job completed before the next operation can be started. Thus it is conceivable that the operation following the one performed on split-mode requires a fairly reasonable processing time that it can be performed on one unit of machine. It means that the split portions of the job would need to be combined into one and moved over to the next machine on which the job's operation is scheduled to be performed.

This research aims at scheduling of jobs with alternative machine options in real time. The processing time of each job would be different on machines of different types that constitute to the unrelated-parallel machining environment. Some machines may turn out to be incapable of processing some jobs in reasonable processing time. Split-jobs are also considered in this research. The number of jobs that needs to be processed in split-modes is pre-determined and not part of the scheduling decision. Each split portion of a job should not be considered as separate jobs, like any other included in the set of jobs to be scheduled. A requirement would need to be imposed to ensure that the difference in completion time of the split portions of the job should be within a user-specified margin. One may even argue that it is perfectly appropriate not to put a constraint on the completion time of the split portions of the job. The reason may be that the split portion completed earlier can be stacked up right by the machine until the other split portion of the job completes its operation on the same machine or another machine in parallel. This line of reasoning is against the underlying concept of Just In Time manufacturing because the portion that is completed earlier has to be carried as work-in-process (WIP) or finished-goods inventory that it cannot be considered for its next operation or be shipped to the customer. In an industry situation, where several jobs compete for the same work center and some or even all of them requiring long processing times, this can mean a long wait of several hours or even days. It means that it is inappropriate to carry a split portion of the job as WIP or finished-goods inventory.

The scheduling environment of this research is dynamic in both job release time and machine availability. However, once a machine becomes available for the first job, it is assumed to be available for the remaining duration of the planning horizon or scheduling time window. The objective of this research focuses on finding the optimal/near-optimal schedule that minimizes the sum of the weighted tardiness of all jobs. Such an objective is important in many industry applications since on-time delivery is one of the most important factors for customer satisfaction. A job can be viewed as a customer order and must be given a 'strategic weight' as a reflection of its priority, i.e. job with higher priority receives higher weight. Tardiness is evaluated as the difference between completion time and due date. If the completion time is less than the due date, the tardiness is counted as zero. Weighted tardiness of a job is calculated as job's weight times its tardiness.

2. LITERATURE REVIEW

Unrelated parallel machines scheduling is the general case of parallel machines scheduling. Identical and uniform parallel machines are two other parallel machining environments. Cheng and Sin (1990) gave a comprehensive review on parallel machines scheduling problems with conventional performance measures based on due date, completion time, and flow time. Alternatively, Lam and Xing (1997) presented a review on parallel machine scheduling problems with non-regular performance measures arising from the concepts of flexible manufacturing systems (FMS) and just-in-time manufacturing (JIT). This review is focused on JIT-oriented criteria, preemption and set up times, and capacitated machines scheduling.

In the past, many efforts have been pursued to identify an efficient scheduling scheme for identical and uniform parallel machining environments. Ho and Chang (1991) proposed a method to minimize the mean tardiness on identical parallel machines. The proposed heuristic used the combination of EDD-SPT dispatching rules and smallest–load machine rule to obtain the initial schedule. The initial schedule was then improved by applying adjacent pairwise interchange technique. Their findings showed that the proposed method performed better than the extension of the algorithm reported previously by Wilkerson and Irwin (1971).

Schutten and Leussink (1996) used a branch-and-bound algorithm to solve identical parallel machines scheduling with dynamic job release dates, general due dates, and family setup times. The objective of the research is to minimize the maximum lateness of all jobs released. The research compared the performance of applying two methods of lower bound to the branch-and-bound algorithm. The first lower bound is based on a method presented by Carlier (1987). The second lower bound is obtained by allowing job preemption. The study concluded that the algorithm using Carlier's lower bound gave the best result.

One of the studies performed on uniform parallel machines scheduling is by Guinet (1995). A heuristic based on simulated annealing is used to solve the uniform parallel machines scheduling problem, which is modeled as a transportation problem in order to minimize the sum of tardiness. The result obtained from the heuristic was compared to a lower bound of the optimal solution. The study suggested that the proposed heuristic gives good results, but only at the expense of a higher computational effort.

Most of the research performed on unrelated parallel machines scheduling was focused on minimizing the maximum completion time, which is also known as makespan. The following investigation was reported for minimizing makespan in an unrelated parallel machining problem. Davis and Jaffe (1979) presented various algorithms that were proven to give a solution that is between $\sqrt{m}$ to $2.5\sqrt{m}$ times the optimum in the worst case. Lenstra et al. (1987) developed an approximation algorithm that guaranteed a makespan that is no longer than twice its optimal. Hariri and Potts (1991) proposed five two-phase heuristics that use linear programming in the first phase to generate a partial schedule, and then apply a heuristic method to schedule the remaining jobs. The study concluded that the quality of the schedules from the two-phase heuristics only is unsatisfactory. Applying either a reassignment heuristic, interchange heuristic, or composite of both further improved the resulting schedule. The improvement heuristics reduced the makespan significantly at a very small computational expense. Suresh and Chaudhuri (1996b) considered a similar problem under two cases of dynamic machine availability: deterministic case and probabilistic case.

Glass et al. (1994), and Piersma and Van Dijk (1996) applied local search heuristics to solve the job-scheduling problem on unrelated parallel machines. The objective is to minimize the maximum completion time. Glass et al. compared three well-known local search methods: simulated annealing, tabu search, and genetic descent algorithm, under an environment of static job release and static machine availability. The performance of each method was tested under two computational time limits: 20 seconds and 100 seconds. This means all methods run for the specified run time, and when the time limit was reached, solutions obtained by each method were collected and compared. Tabu search showed slightly better performance for 20 seconds time limit and there is no significant difference between the three methods for the time limit of 100 seconds. Piersma and Van Dijk proposed a local search algorithm that started by assigning each job to the machine on which it has the shortest processing time (SPT). A job that has

SPT on a machine is referred as job having efficiency value of one on that particular machine. Thus, the starting schedule is a schedule where all jobs have an efficiency of one on the machines they are assigned to. Since this schedule may result in unbalanced workload on the machines, the search procedure is then directed toward the neighborhood of the initial solution to evaluate if any superior solution exists. The neighborhood solutions are obtained by considering schedules that are less 'efficient'. A neighborhood solution is accepted only if it yields a shorter makespan than its parent. The result of the study showed that the performance of the proposed algorithm was generally better than genetic algorithm, simulated annealing and tabu search. The authors also applied the proposed 'efficient' neighborhood search structure to tabu search, which resulted in solutions that are equal to or better than the proposed algorithm.

In an effort to develop optimal schedules for furniture production, Yaghubian et al. (1999) developed a heuristic to solve dry kiln scheduling problem in order to minimize maximum tardiness. The problem is a variant of non-identical parallel machines scheduling problem with dynamic machine availability, limited machine capacity, and transportation time of completed jobs. The effectiveness of the heuristic is compared to the branch-and-bound method. The experimental results indicate that the heuristic is capable of providing high quality solutions in shorter computation time compared to the branch-and-bound method.

Azizoglu and Kirca (1999) approached unrelated parallel machines problems with a general objective that is based on a non-decreasing function of job completion times. They considered total weighted flow time as a special case of this objective function. The authors developed a lower bounding and reduction mechanism that is incorporated into a branch-and-bound algorithm. The performance of the branch-and-bound algorithm with lower bounds was compared to the one without lower bound. The computational experiment indicates that incorporating reduction and bounding scheme significantly improves the performance of the branch-and-bound algorithm.

In real-life, however, it may be desirable to consider a scheduling problem with multiple objectives. Suresh and Chaudhuri (1996a) considered minimizing the maximum tardiness and minimizing the makespan simultaneously on unrelated parallel machines. The proposed algorithm used a heuristic called GAP/EDD, developed by the authors

(1994) to generate an initial solution. This initial solution is used as a starting point for tabu search to obtain alternate solutions. The tabu search employed in this algorithm only utilized the short-term memory feature.

Tabu search has shown remarkable success in solving production-scheduling problems. Barnes et al. (1995) presented an overview of research on production scheduling that applied tabu search. The review listed tabu search-based applications in a single-machine problem, travelling salesman problem, parallel machines problem, flow shop problem, vehicle routing problem, classical job shop problem, and flexible job-shop problem.

Muller et al. (1995) studied the application of tabu search on solving identical parallel machines scheduling problem with sequence dependent setup times to minimize makespan. The algorithm consists of three phases: initial assignment, tabu search, and post-optimization procedure. In the initial stage, each job is assigned to a machine that yields the least increase in completion time. In the tabu search implementation, a neighborhood solution is obtained by removing a job from the busiest machines and inserting it in another machine. A movement that yields the smallest completion time on the busiest processor is applied to the initial solution. This results in a new solution that may or may not be better than previous solution. The process is repeated until a pre-specified total number of iterations without improvement is reached. The search process is then directed to a region that is not explored yet. To employ this diversification strategy, the older job that is less frequently moved in the search process is removed from its machine and inserted in the machine that yields the minimum increase in makespan. The authors compared the performance of this diversification strategy under different values of total number of iterations without improvement.

In parallel-machine scheduling, where the processing time of jobs are extremely unevenly distributed, certain machines may have many short tasks assigned, and others long tasks. For the objective of minimizing the makespan, Hubscher and Glover (1994) applied tabu search with the diversification strategy that seeks to redistribute big jobs and small jobs to every machine. This strategy is based on selecting moves that modify the solution structure influentially. The study concluded that the proposed diversification strategy improved the efficacy of tabu search in obtaining the best solution.

Logendran and Sonthinen (1997) applied tabu search to the job-shop scheduling problem in a flexible manufacturing system, where each part can have more than one process plan and each operation required of a part can be processed on alternative machines. The objective of the study is to minimize the longest completion time for the last operation of all jobs. The authors compared six different versions of tabu search-based heuristics that consisted of all possible combinations of short and long-term memory (using maximal and minimal frequency) with fixed and variable tabu-list sizes. Experimental results concluded that the combination of long-term memory based on maximal frequency and fixed tabu-list size in tabu search is preferred to solve job-shop scheduling problems in flexible manufacturing systems.

3. PROBLEM STATEMENT

Scheduling jobs on multiple machines is not only a matter of sequencing decision such as in single machine scheduling, but also of machine-allocation decision based on machine capability and job characteristics. Referring to the five job characteristics mentioned in Chapter 1, the jobs in this research are characterized by general processing times, general due date (i.e. every job has a different due date), non-preemptive sequencing, independent precedence constraints, and dynamic job release time. This research also considers scheduling split jobs in addition to independent jobs. The need for splitting jobs was explained in Chapter 1. Each split job is assumed to have another split job that comes from the same batch. This means that a batch of job can only be split into two portions. Allowing for larger number of lot splits may result in carrying more work-in-process inventory as lots that are completed earlier have to wait for their split portions in order to move on to the next operation or to be shipped. To maintain a low or zero level of work-in-process inventory, it is necessary to ensure that either the same completion times or a 'small' difference in completion times is identified for the operations representing the two split portions of the job. The difference in completion times of the two split portions of the job must be within a user-specified margin, which in the industry practice is based on some managerial decision. The decision on this margin may be based on inventory capacity or life span of the product.

In addition to above described job characteristics, a 'weight' value is given to each job that represents the priority level of the job. The weight or delay penalty concept was also used by Lee et al. (1997) on a single-machine scheduling problem, and by Vepsalainen and Morton (1987) on a job-shop scheduling problem. In addition to delay penalty, Ow and Morton (1989) also used early job's penalty in their research on single machine scheduling. The objective is to minimize both total earliness and tardiness costs.

The parallel machining environment used in this research is not strictly unrelated, but it is a combination of both unrelated and identical machines. Three levels of machine capability are considered: least, medium and most capable. A problem instance may have more than one unit of machine of the same level of capability. The processing time of a job varies from one machine to another of different level. However, the processing time of the job is the same for machines in the same level. Each machine is available for processing at different times.

The objectives of this research are as follows:

(i) To develop a mathematical model that aims at minimizing the sum of weighted tardiness of all jobs, with some jobs being split-jobs, planned to be scheduled on mixed unrelated – identical parallel machines with dynamic job releases and dynamic machine availability.

(ii) To develop an efficient scheduling algorithm that would solve the model developed in (i).

4. MODEL DEVELOPMENT

4.1. Introduction

The mathematical model for this problem is developed as a mixed integer-linear programming problem. The parameters used in the model such as number of jobs, number of machines, sets of jobs that are split, machine available time, job release time, job weight, job processing time on each machine, job due date, and maximum permissible difference between the completion time of split portions of a job are known quantities.

4.2. Assumptions

(1) The sets of split-jobs are known
(2) Split-jobs that come from the same batch have the same release time, weight, and due date
(3) Setup time is assumed to be included in the processing time
(4) No preemption is allowed
(5) Machine idleness is allowed at no cost
(6) Each machine can only process one job at a time
(7) Typically, a job can be processed on any machine. If a job cannot be processed on a machine, it will have very large processing time on that machine

4.3. Notations

i = 1,2,3, …, m machines
j = 1,2,3, …, n jobs
j' = set of non-split jobs
j'' = set of split jobs.

$j1,j2$ = The first and second split portions of job j, and $(j1,j2) \in j''$

$j = \{j' \cup j''\}$

p_{ij} = processing time of job j on machine i

a_i = time when machine i becomes available

r_j = time when job j is released

w_j = weight assigned to job j

d_j = due date of job j

q_{j1j2} = maximum permissible difference between the completion time of the split portions j1 and j2 of job j

M = an arbitrarily large number

c_{ij} = completion time of job j on machine i

t_{ij} = tardiness of job j on machine i

$$x_{ij} = \begin{cases} 1 & \text{if job } j \text{ is scheduled to be processed on machine } i \\ 0 & \text{otherwise} \end{cases}$$

$$y_{ik\ell} = \begin{cases} 1 & \text{if job } k \text{ precedes job } \ell \text{ on machine } i \\ 0 & \text{otherwise} \end{cases}$$

4.4. Mathematical Model

$$\textit{Minimize} \quad Z = \sum_{j=1}^{n} \sum_{i=1}^{m} w_j t_{ij}$$

subject to:

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad ; j = 1,2,\ldots,n \qquad (1)$$

$$x_{ij}(r_j + p_{ij}) \le c_{ij} \quad ; \begin{array}{l} i = 1,2,...m \\ j = 1,2,...n \end{array} \tag{2}$$

$$x_{ij}(a_i + p_{ij}) \le c_{ij} \quad ; \begin{array}{l} i = 1,2,...m \\ j = 1,2,...n \end{array} \tag{3}$$

$$c_{ij} \le Mx_{ij} \quad ; \begin{array}{l} i = 1,2,...m \\ j = 1,2,...n \end{array} \tag{4}$$

$$c_{i\ell} - c_{ik} + M(1 - y_{ik\ell}) \ge x_{i\ell} p_{i\ell} \quad ; \begin{array}{l} i = 1,2,...m \\ k,\ell = 1,2,...n \ (k < \ell) \end{array} \tag{5}$$

$$c_{ik} - c_{i\ell} + My_{ik\ell} \ge x_{ik} p_{ik} \quad ; \begin{array}{l} i = 1,2,...m \\ k,\ell = 1,2,...n \ (k < \ell) \end{array} \tag{6}$$

$$c_{gj1} - c_{hj2} \le q_{j1j2} + M(2 - x_{gj1} - x_{hj2}) \quad ; \begin{array}{l} g,h = 1,2,...m \\ (j1,j2) \in j" \end{array} \tag{7}$$

$$c_{hj2} - c_{gj1} \le q_{j1j2} + M(2 - x_{gj1} - x_{hj2}) \quad ; \begin{array}{l} g,h = 1,2,...m \\ (j1,j2) \in j" \end{array} \tag{8}$$

$$c_{ij} - d_j \le t_{ij} \quad ; \begin{array}{l} i = 1,2,...m \\ j = 1,2,...n \end{array} \tag{9}$$

$$t_{ij} \ge 0 \quad ; \begin{array}{l} i = 1,2,...m \\ j = 1,2,...n \end{array} \tag{10}$$

4.5. Model Description

The mathematical model developed above is a mixed (binary) integer-linear programming model as both real and binary integer (0/1) variables are included. The objective function of the model focuses on minimizing the sum of the weighted tardiness of all jobs released. Constraint (1) states that the operation required of each job, either split or non-split, is performed on only one machine. As machine availability and job release time are dynamic, the earliest start time of a job must be the largest of the job release time and the availability time of the machine to which the job is assigned. Constraints (2) and (3) jointly ensure that a job's completion time is at least equal to or greater than the sum of its earliest start time and its processing time. Constraint (4) states that the completion time of a job on a machine is zero if its operation is not performed on that machine. Constraints (5) and (6) jointly assure that two jobs are not processed at the same time on a machine. Constraints (7) and (8) jointly ensure that the absolute difference between the completion time of the split portions, j1 and j2, of job j is less than or equal to the permissible maximum, q_{j1j2}. These constraints are binding only when both binary integer variables, x_{gj1} and x_{hj2}, equal to one. If the constraints are binding, the absolute difference between the completion time of j1 and j2 will be less than or equal to q_{j1j2}. Constraint (9) ensures that the tardiness of a job is greater than or equal to the difference between its completion time and due date. Finally, constraint (10) guarantees that only positive values for tardiness are considered.

4.6. Computational Complexity of the Research Problem

The computational complexity of the research problem can be determined from considering a special case in total weighted tardiness problem. Lenstra et al. (1977) proved that single-machine scheduling problem with all jobs' weight being equal is NP-hard in the strong sense. Single-machine problem with equal weight of jobs is a special case of unrelated parallel machines with general weight of jobs. If the special case of the research problem is strongly NP-hard, then the research problem must be strongly NP-hard as well. If a problem is NP-hard, it is very unlikely that its optimal solution can be found in polynomial time.

An implicit enumeration method such as the branch and bound technique can only be used to solve small problem instances in reasonable computational time. For medium and large problem instances, the branch and bound technique would not only be highly time consuming, but in some cases may never find the optimal solution even after an exceedingly large computational time. Thus, there is a need for developing a better methodology that yields an optimal/near-optimal solution fairly efficiently, especially for medium and large problem instances. One of the higher-level search heuristics that has been applied to solve production-scheduling problems is tabu search. Barnes et al.

(1995) presented a review of tabu search application to various machine scheduling problems such as single machine, parallel machines, open shop, flow shop and job shop. Tabu search-based heuristics have also been applied to scheduling problems with different objectives on unrelated parallel machines (Glass et al., 1994, Piersma and Van Dijk, 1996, and Suresh and Chaudhuri, 1996a).

5. HEURISTIC ALGORITHM

5.1. Introduction

The heuristic algorithm utilizes tabu search as the mechanism to explore the solution space. Tabu search was first introduced by Glover (1986). It is a strategic heuristic procedure for solving combinatorial optimization problems. It is designed to overcome the limitation of local optimality that is frequently encountered by other methods. Tabu search can guide the search process from one solution state to another by strategically constraining and freeing the attributes of the search process. This is possible because tabu search uses flexible memory functions that record search information of varying time spans. The long-term memory functions can be used to intensify the search by reinforcing attributes that are historically found good, or diversify the search to unexplored regions.

The information about tabu search in detail, including fundamental principles, advanced settings and guidelines can be found in Glover (1989, 1990a and 1990b). The mechanism of tabu search is explained in the next section. Then, four methods to obtain the initial solution are presented, followed by generation of neighborhood solutions and steps of tabu search. Finally, an example problem is used to show the application of the heuristic algorithm.

5.2. Tabu Search Mechanism

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

1. The use of flexible memory structures to store information during the search process. It allows the evaluation criteria and historical search information to be exploited more thoroughly than by rigid memory structures (as in branch-and-bound) or by memoryless systems (as in simulated annealing and other randomized approaches).

2. A control mechanism that is based on the interplay between imposing and freeing the constraints on 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.

The approach taken by tabu search is similar to the hill-climbing heuristic. It directs the search progressively from an initial solution to a better one in an upward manner for maximizing the objective function. In minimization context, the hill is inverted and the direction is downward. The limitation of hill-climbing heuristic is that the optimum obtained at the end of the search is a local optimum, which may not be the global optimum. Tabu search is capable of guiding such a heuristic away from being trapped at a local optimum and to continue the exploration to reach a global optimum or near global optimum.

Tabu search always begins with an initial solution. This initial solution can be randomly or systematically generated. It can be a feasible or an infeasible solution. However, starting the search with a 'good' feasible solution may speed up the process to get to an optimal/near-optimal solution. This is because the solution space is wider if the search process starts from an inferior initial solution. The wider the solution space is, the longer it takes to get to an optimal/near-optimal solution. Consequently, having an infeasible solution as an initial solution may prolong the computation time needed by tabu search to get to an optimal/near-optimal solution. Since a good initial solution is important for tabu search, four different methods for generating initial solutions are developed. These methods are explained in detail in the next section.

Having the initial solution in hand, one can go about exploring the solutions in the neighborhood by perturbing the initial solution. Every neighborhood solution is evaluated by a performance criterion, which in this research is the total weighted tardiness. A neighborhood solution will be considered admissible if the move that yields the solution passes a tabu-status check. The primary goal of the tabu restriction is to permit the search process to go beyond points of local optimality while still making high quality moves at each step. The tabu restriction is embodied in tabu list. The tabu list consists of the changes or moves recently applied in order to direct one state of solution to another. It also records recent moves in the order in which they are made. The length of time a tabu move is enforced depends on the size of tabu list. Past research has shown that tabu-list size depends on the size of the problems being investigated. Thus, a prior experimentation is required to determine a good size for the tabu list.

By restricting the search to moves that are not tabu, the search process is prevented from revisiting solutions found earlier. However, a tabu move may yield a better solution than the one found so far. Therefore, an aspiration criterion is used to counterbalance tabu restrictions. This means that the tabu restriction can be overridden if an aspiration criterion is satisfied. The aspiration criterion gives a tabu move a second chance to be considered in the search process. After all neighborhood solutions are tested against tabu status and aspiration criteria, the move that yields the best solution is selected for future perturbation. This solution is admitted to the candidate list (CL). The whole process is then repeated until certain criterion is satisfied to terminate the search. Every chosen best solution has to be checked against the CL. This check is necessary to assure that a solution is not considered more than once for perturbation.

There are different schemes to terminate the search process. One way is to let the process run until a certain size of the CL is achieved. Another method is to let the process run up to a certain number of consecutive iterations that do not yield any improvement. Yet another method is to impose a limit on the computation time used in the search process.

Essentially, tabu list is the short-term memory of tabu search. The effect of short-term memory can be amplified by applying the long-term memory function. The long-term memory can be used to direct the search to focus on the region that is historically found good (intensification process) or on the region that is hardly visited (diversification process). The long-term memory is embodied in a frequency matrix that keeps track of the essential information of all previous moves. A new starting point can be identified using the information from long-term memory. The search process will use this starting point as an initial solution to do a restart.

5.3. Initial Solution

Over the years, researchers have tried to use simple rules to solve tardiness problems, which include total tardiness, weighted tardiness and maximum tardiness problems. A number of simple dispatching rules such as the Earliest Due Date (EDD), Shortest Processing Time (SPT), Minimum Slack (MSLACK) and Slack per Remaining Processing Time (S/RPT) have been applied to solve these problems. The first two rules are time-independent, meaning that the job priority is dependent on job and machine data, and remains the same throughout the scheduling horizon. Contrastingly, the last two dispatching rules are time-dependent, i.e. job priority is dependent on the time when machines become available after processing the preceding job. In his survey on the total tardiness problem, Koulamas (1994) compiled several research efforts that have used simple dispatching rules to sort jobs to be allocated to the available machines or to construct an initial schedule, which is further improved by applying heuristic methods.

While simple dispatching rules only use a single attribute to achieve its objective, in industry practice, there is more than one attribute that determines a 'good' schedule. Attribute is 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. A composite dispatching rule is designed to combine several job and machine attributes to obtain a good schedule. It is a function made up of attributes and some scaling parameters. A number of composite dispatching rules have been developed for different types of machine environments. These include Dynamic Composite Rule/DCR (Conway et al., 1967), Cost Over Time/COVERT (Carroll, 1965), Apparent Tardiness Cost/ATC (Vepsalainen and Morton, 1987), and Apparent Tardiness Cost with Setup/ATCS (Lee et al., 1997).

In this research, four different methods are used to generate the initial solution for tabu search. Two of them are based on EDD. One method is based on a combination of Least Flexible Job (LFJ) and Least Flexible Machine (LFM) rules. The last method is a modified version of ATC that incorporates dynamic job release time. The following notations will be used throughout the development of the algorithm:

t = clock time

i = machine index

j = job index

a_i = initial availability time of machine i

mt_i = release time of machine i (after processing a job)

NS = set of unscheduled jobs

i* = selected machine index

j* = selected job index

q_{j1j2} = maximum permissible difference between the completion time of split portions j1 and j2 of job j

p_{ij} = processing time of job j on machine i

r_j = release time of job j

CT(j,i) = completion time of job j on machine i

ST(j,i) = starting time of job j on machine i

5.3.1. Earliest Due Date (EDD)

The following steps are developed based on the EDD rule and used to generate the initial solution:

1. Initially, set t = 0 and $mt_i = a_i \ \forall$ i. Include all jobs into NS.
2. Select the machine/unit (i) that has the minimum mt_i. If there is more than one machine with minimum mt_i, break ties by choosing the machine with the smallest index. Let the selected machine be i*. Set $t = mt_{i*}$.
3. Let SJ = the set of jobs released at or earlier than t, and that can be processed on i* ($SJ \subset NS$).
    a. If $SJ = \varnothing$, find a job/jobs from NS that can be processed on i* and has/have minimum r_j.
        i. If all jobs in NS cannot be processed on i*, exclude machine i* from future consideration. Go to step 6.
        ii. If only one job is found, select this job and assign it to i*. Go to step 4.

iii. If two or more jobs are found, break ties in favor of the EDD rule, followed by the highest weight. If job ties still exist, check if a pair of split jobs are among the competing jobs. If a pair of split jobs are among the competing jobs, select the split portion that has the largest p_{i*j} to i*. Otherwise, assign the job with the smallest index to i*. Go to step 4.

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

c. If SJ has two or more jobs, check if any of the jobs in SJ is a split job with its split portion eliminated from SJ (i.e. its split portion was scheduled).

i. If such a job exists, assign it to i*. Go to step 4.

ii. If two or more such jobs exist, the priority is given to the split job that has its split portion eliminated earliest from SJ. Assign this split job to i*. Go to step 4.

iii. If such a job does not exist, choose the job with the EDD from the SJ list. Break ties by choosing the job with the highest weight. If more than one job has the EDD and highest weight, check the split status of these jobs. If a pair of split jobs are among the competing jobs, choose the split job portion with the largest p_{i*j}; otherwise, choose the job with the smallest index. Assign the selected job to i*. Go to step 4.

4. Set j* = the selected job; ST(j*,i*) = max [mt_{i*}, r_{j*}] and CT(j*,i*) = ST(j*,i*) + p_{i*j*}. If j* is a non-split job or a split job that has its other split portion unscheduled, go to step 5. If j* is a split job and its split portion, denoted by j'*, was scheduled, let i'* be the machine on which j'* was scheduled. Check the difference between CT(j*,i*) and CT(j'*,i'*). If CT(j'*,i'*) - CT(j*,i*) > q_{j1j2}, set CT(j*,i*) = CT(j'*,i'*) - q_{j1j2} and ST(j*,i*) = CT(j*,i*) - p_{i*j*}.
5. Set mt_{i*} = CT(j*,i*) . Eliminate j* from NS.
6. If NS $\neq \varnothing$, go to step 2.

The algorithmic procedure in step 4 needs further explanation, specifically for split jobs. In general, a split job (j*) that has its split portion (j'*) previously scheduled will fall under any one of the following three cases: 1.) CT(j'*,i'*) - CT(j*,i*) > q_{j1j2}; 2.) CT(j*,i*) - CT(j'*,i'*) > q_{j1j2}; and 3.) |CT(j'*,i'*) - CT(j*,i*)| $\leq q_{j1j2}$. An adjustment in the completion time of a split portion is required for cases 1 and 2 to make the solution feasible (i.e. to satisfy the JIT constraints). In case 1, the adjustment is made by delaying the start time of j* so that the difference between the completion times of j* and j'* is at most equal to q_{j1j2}. In case 2, however, an adjustment in completion time cannot be made since j'* was previously scheduled and thus, its start time and completion time are considered permanent. Under case 2, the algorithm would identify an infeasible initial solution. Starting the tabu search with an infeasible initial solution may prolong the computation time required to find an optimal/near-optimal solution. This issue was previously discussed in section 5.2.

5.3.2. Earliest Due Date with consideration for split jobs (EDDsp)

The scheduling steps using EDDsp are somewhat similar to EDD. The difference is in the way the split jobs are scheduled. In EDDsp, immediately after a split portion of a job is scheduled, the algorithm will assign its other split portion to the machine that can complete it earliest. This is done to ensure that the JIT constraints for split portions of the same job are satisfied. This means that the initial solution is guaranteed to be feasible. The steps associated with the algorithm can be presented as follows:

1. Initially, set $t = 0$ and $mt_i = a_i \ \forall\ i$. Include all jobs into NS.
2. Select the machine/unit (i) that has the minimum mt_i. If there is more than one machine with minimum mt_i, break ties by choosing the machine with the smallest index. Let the selected machine be i*. Set $t = mt_{i*}$.
3. Let SJ = the set of jobs released at or earlier than t, and that can be processed on i* ($SJ \subset NS$).
   a. If $SJ = \varnothing$, find a job/jobs from NS that can be processed on i* and has/have minimum r_j.
      i. If all jobs in NS cannot be processed on i*, exclude machine i* from future consideration (i.e. machine i* will be no longer considered as one of the available machines). Go to step 6.
      ii. If only one job is found, select this job and assign it to i*. Go to step 4.

iii. If two or more jobs are found, break ties in favor of the EDD rule, followed by the highest weight. If job ties still exist, check if a pair of split jobs is among the competing jobs. If a pair of split jobs are among the competing jobs, select the split portion that has the largest p_{i*j} to i*. Otherwise, assign the job with the smallest index to i*. Go to step 4.

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

c. If SJ has two or more jobs, break ties in favor of the EDD rule, followed by the highest weight. If more than one job has the EDD and highest weight, check the split status of these jobs. If a pair of split jobs are among the competing jobs, select the split portion of the job that has the largest p_{i*j} to i*. Otherwise, assign the job with the smallest index to i*. Go to step 4.

4. Let the selected job be j*; ST(j*,i*) = max [mt_{i*}, r_{j*}], CT(j*,i*) = ST(j*,i*) + p_{i*j*}, and mt_{i*} = CT(j*,i*). If j* is a non-split job, go to step 5. If j* is a split job, let its split portion be j'*. Let SM be a set of machines that can process j'*. Calculate the starting and completion time of j'* on each machine included in SM: ST(j'*,k) = max [mt_k, $r_{j'*}$], CT(j'*,k) = ST(j'*,k) + $p_{kj'*}$, $k \in$ SM. Select the machine that can complete j'* the earliest and call this machine i'*. Let CT(j'*,i'*) and ST(j'*,i'*) be the completion and start time of j'* on i'*, respectively. Check the difference between CT(j*,i*) and CT(j'*,i'*):

a. If CT(j'*,i'*) - CT(j*,i*) > q_{j1j2}, set CT(j*,i*) = CT(j'*,i'*) - q_{j1j2}, ST(j*,i*) = CT(j*,i*) – p_{i*j*}. Set mt_{i*} = CT(j*,i*) and $mt_{i'*}$ = CT(j'*,i'*). Eliminate j* and j'* from NS. Go to step 6.

b. If CT(j*,i*) - CT(j'*,i'*) > q_{j1j2}, set CT(j'*,i'*) = CT(j*,i*) - q_{j1j2}, ST(j'*,i'*) = CT(j'*,i'*) - $p_{i'*j'*}$. Set $mt_{i'*}$ = CT(j'*,i'*). Eliminate j* and j'* from NS. Go to step 6.

c. If |CT(j*,i*) - CT(j'*,i'*)| $\leq q_{j1j2}$, set $mt_{i'*}$ = CT(j'*,i'*). Eliminate j* and j'* from NS. Go to step 6.

Note: The adjustment in completion times of the split portions of a job to meet JIT requirement, if needed, is made only by delaying the start time of a split portion. One may consider starting a split portion earlier rather than delaying the start time as another way of adjusting the completions times to meet the JIT requirement. The adjustment done by this way may cause changes on the start and completion times of the jobs that were previously scheduled. This possibility is ignored since all jobs that were previously scheduled are considered to have permanent start time and completion time.

5. Eliminate j* from NS.
6. If NS $\neq \varnothing$, go to step 2.

Step 4 shows the primary difference between EDD and EDDsp. EDDsp method assures that the absolute difference in completion times between the split portions of a job is at most equal to q_{j1j2}. Thus, this method will surely identify a feasible initial solution.

5.3.3. Least Flexible Job and Least Flexible Machine (LFJ/LFM)

Centeno and Armacost (1997) developed an algorithm for scheduling jobs in parallel machines with dynamic job release time, due dates, and different machine capability in order to minimize the maximum lateness. The lateness of a job is different from the tardiness. Lateness is evaluated as completion time minus due date, thus it may be a negative, zero or positive value. On the other hand, tardiness can either be zero or positive value. Machine capability is reflected in the number of jobs the machine is capable of processing, i.e. the most capable machine has the potential to process the most number of jobs. The job due date is generated as the job release time plus a constant. The algorithm is based on Least Flexible Job (LFJ) and Least Flexible Machine (LFM) rules. LFJ is defined as the job that can be processed by the least number of machines. LFM is the machine that is capable of processing the least number of jobs. The LFJ rule gives higher priority to less flexible jobs and thus, prevents them from being late due to their inflexibility. The LFM rule ensures that less capable machines get a fair share of assignment as more capable machine.

A method to generate the initial solution is developed based on Centeno and Armacost's algorithm. This method uses the same mechanism as that in EDDsp to schedule split jobs and make the adjustment to the completion times of the split portions of a job. The steps associated with this method can be documented as follows:

1. Initially, set $t = 0$, $mt_i = a_i \ \forall\ i$ and include all jobs in NS.
2. Check if any $r_j \leq t$. If yes, then go to step 4.
3. Set $t = \min [r_j]$ where $j \in$ NS.
4. Choose the least flexible job with $r_j \leq t$. If two or more jobs are chosen, select the job with min r_j. If two or more jobs with min r_j are selected, do one of the following:
   a. If the split jobs are among the selected jobs, choose the pair of split jobs with the smallest indices. Go to Step 6.
   b. If all selected jobs are non-split, choose the job with the smallest index. Go to step 5.
5. Let the selected job be j*. Find the least flexible machine that can process j* and is currently idle. If two or more machines are found, break ties by selecting the machine with the smallest index. If all capable machines are busy, then go to step 8. Let the selected machine be i*. Set $ST(j^*,i^*) = \max [t, mt_{i^*}]$, $CT(j^*,i^*) = ST(j^*,i^*) + p_{i^*j^*}$ and $mt_{i^*} = CT(j^*,i^*)$. Eliminate j* from NS and go to step 7.
6. Find the least flexible machine that is capable of processing the selected pair of split jobs (a machine that is capable of processing one split portion of a job should be able to process the other split portion), and is currently idle. If two or more machines are found, break ties by selecting the machine with the smallest index. If all capable machines are busy, then go to step 8. Let the selected machine be i*. From the two split portions, let the split portion that has the larger processing time on i* be j*, and the other split portion be j'*. Assign j* to i* and set $ST(j^*,i^*) = \max [t,mt_{i^*}]$, $CT(j^*,i^*) = ST(j^*,i^*) + p_{i^*j^*}$ and $mt_{i^*} = CT(j^*,i^*)$. Let SM be the set of machines that can process j'*. Calculate the starting and completion time of j'* on each machine included in SM: $ST(j'^*,k) = \max [mt_k, r_{j'^*}]$, $CT(j'^*,k) = ST(j'^*,k) + p_{kj'^*}$, $k \in$ SM. Select the machine that can complete j'* the earliest and call this machine i'*. Let $CT(j'^*,i'^*)$ and $ST(j'^*,i'^*)$ be the completion and start time of j'* on i'*, respectively. Check the difference between $CT(j^*,i^*)$ and $CT(j'^*,i'^*)$:

a. If $CT(j'^*,i'^*) - CT(j^*,i^*) > q_{j1j2}$, set $CT(j^*,i^*) = CT(j'^*,i'^*) - q_{j1j2}$, $ST(j^*,i^*) = CT(j^*,i^*) - p_{i^*j^*}$. Set $mt_{i^*} = CT(j^*,i^*)$ and $mt_{i'^*} = CT(j'^*,i'^*)$. Eliminate j* and j'* from NS. Go to step 7.

b. If $CT(j^*,i^*) - CT(j'^*,i'^*) > q_{j1j2}$, set $CT(j'^*,i'^*) = CT(j^*,i^*) - q_{j1j2}$, $ST(j'^*,i'^*) = CT(j'^*,i'^*) - p_{i'^*j'^*}$. Set $mt_{i'^*} = CT(j'^*,i'^*)$. Eliminate j* and j'* from NS. Go to step 7.

c. If $|CT(j^*,i^*) - CT(j'^*,i'^*)| \leq q_{j1j2}$, set $mt_{i'^*} = CT(j'^*,i'^*)$. Eliminate j* and j'* from NS. Go to step 7.

7. Set newt = min $[mt_i]$ $\forall$ i and t = max [t,newt]. Go to step 9.
8. Set newt = min $[mt_i]$, for all machine i that are capable of processing j* (from step 5) or the selected pair of split jobs (from step 6). Set t = newt and go to step 2.
9. If NS $\neq \varnothing$, go to step 2; otherwise, stop.

5.3.4. Apparent Tardiness Cost (ATC)

Rachamadugu and Morton (1981) developed a look-ahead rule for the single machine weighted tardiness problem. This heuristic uses the composite dispatching rule to calculate the Apparent Priority (AP) for all unscheduled jobs. By extending the use of AP rule, Vepsalainen and Morton (1987) developed the Apparent Tardiness Cost (ATC) rule to schedule jobs in job shop environment in order to minimize the total weighted tardiness. Lee et al. (1997) further applied the ATC rule to the single machine scheduling problem with sequence–dependent setup time in order to minimize the total weighted tardiness. This rule is called Apparent Tardiness Cost with Setups (ATCS) rule.

ATC rule calculates the priority index for all unscheduled jobs at any instant when a machine is free. The priority function is evaluated as:

$$PI_j(t) = \frac{w_j}{p_j}\exp\left[-\frac{\max[d_j - p_j - t, 0]}{k\bar{p}}\right]$$

where t is the time when the machine is available; w_j, p_j, d_j are the weight, processing time and due date of job j, respectively; $\bar{p}$ is the average processing time of all remaining unscheduled jobs, and k is the look-ahead parameter. This function consists of two components: w_j/p_j and the exponential term. The first component gives a high priority to the job that has small processing time and large weight. Thus, it represents the Weighted Shortest Processing Time (WSPT) dispatching rule. The numerator in the second component is max $(d_j-p_j-t, 0)$, which can be translated as the slack for job j at time t. In general, this component gives a high priority to jobs with small slack. Thus, it represents the least-slack dispatching rule (LSR). The parameter k provides the look-ahead capabilities of the ATC rule. It means that parameter k carries the value, which represents the shop congestion level. It is related to the number of competing jobs (Rachamadugu and Morton, 1981). In their experimental study, Lee et al. (1997) have shown that this parameter depends on the characteristics of the problem instance.

In order to apply the ATC rule to unrelated parallel machines with dynamic machine availability and dynamic job releases, the priority function should be changed. The processing time of job j on machine i, p_{ij}, must be used instead of p_j in the function as this research deals with an unrelated parallel machining environment. No changes are necessary to incorporate dynamic machine availability, as in the original application of ATC, the priority index is evaluated for all unscheduled jobs whenever a machine becomes available. The applications of ATC rule in previous studies assume static job releases. For dynamic job release time, Lee et al. (1997) suggested that the job selected (based on the priority index) should be among the released jobs. This practice prevents unreleased jobs from competing with released jobs for the available machine. A different mechanism should be developed to provide all jobs (released or not) the same opportunity to compete for the available machine. It should be reflected in the priority index. Therefore, an additional component that considers job release time is included in the priority function. Three models that use job release time were tested:

1. $\exp\left[-\frac{\max[r_j - t, 0]}{k\bar{r}}\right]$
2. $\exp\left[-\frac{r_j - t}{k\bar{r}}\right]$
3. $\exp\left[-\frac{r_j}{k\bar{r}}\right]$

where r_j is the release time of job j; t is the time when the machine is available; $\bar{r}$ is the average release time of the remaining unscheduled jobs, and k is a look-ahead parameter.

In the first model, job release time influences the priority index only if the job release time is larger than the current time, t. In other words, a job that was released prior to or at t will get its priority value from other components of ATC. On the contrary, in the second and third models, job release time influences the priority index regardless of when the job is released, i.e. before or after t. In the second model, the impact of job release time on the priority index is amplified by t. The priority index will increase exponentially if t is larger than the job release time. In the third model, the impact of the job release time on the priority index is not influenced by t. All three models give high priority to a job with small release time.

Intuitively, the first model serves the objective of minimizing the total weighted tardiness better than the other models. Once a job is released, the criticality of the job should be determined by the weight of the job and how close the job is to its due date. The time span after the job is released (i.e. the waiting time of the job on the shop floor) should not matter as much as meeting the due date. In the second and third models, the waiting time is considered along with the weight and slackness of the job. In order to compare the performance of the three models, 15 problem instances were generated. These problem instances have different total number of jobs and machines. The processing time, release time, weight, and due date for each job, and the machine release time are generated by randomization procedure. Each model is then applied to all problem instances. The test results show that the first model yields the best solution 62% of the time, second model 15%, and third model 23%. Thus, the first model is used to accurately depict the priority function. The priority function selected for the scheduling problem on unrelated parallel machines with dynamic job release and dynamic machine availability is:

$$PI_j(t_i) = \frac{w_j}{p_{ij}} \exp\left[-\frac{\max[d_j - p_{ij} - t_i, 0]}{k_1 \bar{p}_i}\right] \exp\left[-\frac{\max[r_j - t_i, 0]}{k_2 \bar{r}}\right]$$

PI_j = the priority index of job j

t_i = time when machine i is available

p_{ij} = processing time of job j on machine i

$\bar{p}_i$ = average processing time of the remaining unscheduled jobs that can be processed on machine i

w_j, r_j, d_j = weight, release time, and due date of job j, respectively

$\bar{r}$ = average release time of the remaining unscheduled jobs that can be processed on machine i

k_1, k_2 = look-ahead parameters

It is important to use the appropriate values of k_1 and k_2 as they represent the look-ahead capability of the ATC rule. The values of the parameters k_1 and k_2 depend on the problem instance as they essentially perform scaling for shop congestion level of the problem instance. Therefore, before any explanation on these parameters is given, it is important to know the factors that characterize a problem instance. Four different factors are used to characterize a problem instance in this research; they are number of jobs (J), number of machines (M), tardiness factor (τ) and due date range factor (R). The last two factors have been used before by Lee et al. (1997), Suresh and Chaudhuri (1994), and Ow and Morton (1989) to determine the due date of a job. The tardiness factor, τ, is defined as $\tau = 1 - \bar{d} / C_{max}$ where $\bar{d}$ is the average due date and C_{max} is the maximum completion time of all jobs released (makespan). Ow and Morton (1989) state that "τ is a coarse measure of the proportion of jobs that might be expected to be tardy in an arbitrary sequence". A large value of τ means that the average due date is much smaller than C_{max}, which implies that the completion times of most jobs are bound to exceed their respective due dates. On the other hand, a small value of τ implies that most jobs are bound to complete before their respective due dates. Thus, large value of τ indicates tight due dates and small τ loose due dates. The due date range factor, R, is defined as $R = (d_{max} - d_{min}) / C_{max}$ where d_{max} and d_{min} are the maximum and minimum due date, respectively. R provides the measure of variability of the due dates. Different combinations of τ and R generate due dates with various characteristics such as those shown in Table 5.1 (Suresh and Chaudhuri, 1994).

τ, R and C_{max} have to be used simultaneously in order to generate due dates of jobs. However, C_{max} is schedule-dependent. An estimation scheme has to be developed for C_{max}. Given the job processing time, job release time and machine availability time, the C_{max} of the problem can be estimated as:

$$C_{max} = \begin{cases} \dfrac{\sum_{j=1}^{n} \dfrac{\sum_{i=1}^{m_j} \max[r_j, a_i] + p_{ij}}{m_j}}{m} & \text{if } n \geq m \\ \dfrac{\sum_{j=1}^{n} \dfrac{\sum_{i=1}^{m_j} \max[r_j, a_i] + p_{ij}}{m_j}}{n} & \text{otherwise} \end{cases}$$

where n = total number of jobs; m = total number of machines; m_j = total number of machines that can process job j; r_j = release time of job j; a_i = availability time of machine i; p_{ij} = processing time of job j on machine i

Table 5.1 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

The appropriate values for k_1 and k_2 have to be selected for ATC rule to be an effective dispatching rule. Since k_1 and k_2 are predicted to be problem dependent, it is necessary to find the values of k_1 and k_2 as a function of J, M, τ, and R. An experimental study similar to the one conducted by Lee et al. (1997) is performed to determine k_1 and k_2 for the priority function used in this research. This experimental study is conducted to investigate $k_1 = f_1$ (J, M, τ, R) and $k_2 = f_2$ (J, M, τ, R). The experiment is conducted over five values of job (J = 5, 20, 35, 55, 70), four values of machines (M = 3, 8, 14, 20), five values of τ (τ = 0.2, 0.35, 0.5, 0.65, 0.8), and four values of R (R = 0.2, 0.4, 0.7, 1.0). A problem type is determined by a combination of these four factors. Thus, there are a total of 400 (5*4*5*4) unique problem types. Within each problem type, two problem instances are generated by using different random number seeds, resulting in a total of 800 problem instances.

From the total number of jobs, J, 25% of the jobs are assumed to be split jobs. Since split jobs have to be in pairs, the total number of split jobs should be even. If 25%*J results in a decimal value, the value rounded to the nearest even number is used. If 25%*J results in an odd number, the value rounded up to the nearest even number is used. For example, if J = 35, the total number of split jobs will be 10 or 5 pairs of split jobs. The split status is randomly assigned to the jobs until the total number of split jobs is reached. The machines can be grouped into three levels of capability: least, medium and most capable machines. All three types of machines are included in each problem instance. Each machine type may have more than one unit. The sum of machine units of all machine types is equal to M. The least, medium and most capable machines have the potential to process 50%, 70% and 85% of all jobs, respectively. Each machine type is assigned a coefficient of capability, α_m, which is uniformly distributed over the interval [1,10]. Thus, three values are needed for α_m, one for each machine type. The largest value is given to the least capable machine and smallest to the most capable machine. This coefficient is used to generate job-processing times. The processing times are uniformly distributed over the interval $[\alpha_m + 1, \alpha_m + 20]$ for non-split jobs and $[\alpha_m + 11, \alpha_m + 20]$ for split jobs. The processing times of a job are the same for machines of the same type. Job release time and machine availability time are generated from a Poisson distribution (Schutten and Leussink, 1996, and Suresh and Chudhuri, 1996b) with parameter $\lambda = 5$. Job weight is uniformly distributed over the interval [1,4]. The due dates are generated from a composite uniform distribution based on R and τ (Lee et al., 1997). With probability τ the due date is uniformly distributed over the interval $[\bar{d} - R\bar{d}, \bar{d}]$ and with probability (1-τ) over the interval $[\bar{d}, \bar{d} + (C_{max} - \bar{d})R]$.

For each problem instance, the ATC rule is applied repeatedly using different combinations of values for k_1 and k_2. The values of k_1 tested range from 0.2 to 8.0 with an increment of 0.2, and the values of k_2 tested range from 0.2 to 6.0 with an increment of 0.2. Thus, there are a total of 1200 combinations of k_1 and k_2 values. *For each problem instance*, the values of the total weighted tardiness (TWT) were evaluated as a result of applying the ATC rule repeatedly using 1200 combinations of k_1 and k_2. All k_1 values that yielded the minimum total weighted tardiness (MTWT) are identified. The average of these k_1 values is referred to as $\bar{k}_1$. The average of all k_2 values that yielded the minimum total weighted tardiness is collected the same way as k_1, and it is referred to as $\bar{k}_2$. A range of k_1 values is then selected by considering the entire range of k_1 (0.2, 0.4, 0.6, …8.0) with $\bar{k}_2$ that resulted in total weighted tardiness that is less than or equal to MTWT(1+β). The parameter β is a tolerance value for MTWT that is a decreasing function of τ and ranges from 0 to 0.065 (Lee et al., 1997). The value of 0.065 (6.5%) is the upper limit of the tolerance value for MTWT. Although the scheduling problem that was addressed by Lee et al. is different from the problem addressed in this research, the upper limit of β that they used is very reasonable. It provides sufficient flexibility for selecting an appropriate range of k_1 values. The midpoint of the selected range of k_1 is used as the recommended value for k_1. Similarly, a range of k_2 values is selected by considering the entire range of k_2 (0.2, 0.4, 0.6, …6.0) with $\bar{k}_1$ that resulted in total weighted tardiness that is less than or equal to MTWT(1+β). The midpoint of the selected range of k_2 is used as the recommended value for k_2. Therefore, each problem instance has a pair of recommended values for k_1 and k_2. The entire procedure is repeated until all 800 problem instances are tested.

In an initial data exploration, the experimental results for the recommended values of k_1 appeared to form two clusters, one cluster around the smaller values of k_1 (i.e. 0 to 2.5) and the second one ranges from 3.9 to 4.5 with a concentration at 4.1. A preliminary investigation is done to explore the cause of these clusters. This leads to checking why a number of data points are concentrated at 4.1. The investigation is carried back to the range of values used to identify the recommended k_1 as a midpoint. It appeared that the entire range of values between 0.2 and 8.0 contributed to the recommended k_1 of 4.1.

Recall from the testing procedure, the range of k_1 is selected by considering the entire range of k_1 (0.2, 0.4, 0.6, ...8.0) with $\bar{k}_2$ that resulted in total weighted tardiness that is less than or equal to MTWT(1+β). The implication of the entire range being selected is that each value from 0.2 to 8.0 yields the same TWT. This implies that the problem instances with recommended k_1 equals to 4.1 are not responsive to different values of k_1 used in the ATC rule. This means that applying the ATC rule to these problem instances will yield the same total weighted tardiness regardless of the value of k_1 used. Thus, there is no need to find the function that relates k_1 to the values of the factors (J, M, τ, R) that correspond to these problem instances. A further analysis shows that there is a commonality in the ratio of J/M between the problem instances with recommended k_1 of 4.1. Generally, when J/M ratio of the problem instances is less than 1.7, their recommended k_1 is approximately equal to 4.1. This means that problem instances with J/M ratio less than 1.7 are generally not responsive to the changes in k_1.

Excluding all data points that have J/M ratios less than 1.7, a multiple linear regression analysis is conducted on the remaining data. In the first attempt, k_1 is fitted on a simple model that includes only all main effects (J, M,τ, R). Using a test significance level (α) of 0.05, it turned out that factor J does not have a statistically significant effect on k_1. Then, excluding factor J, k_1 is fitted to a richer model that included the main effects M, τ, R, and all possible interaction terms between them. The significant interaction terms are M*τ and R*τ. Lastly, k_1 is fitted to an even richer model that included M, τ, R, M*τ, R*τ, the quadratic terms and cubic terms of the main effects. Only the quadratic term of τ showed any statistical significance and none of the cubic terms were significant. When the interaction of R and τ is included in the model, the main effect R turns out to be insignificant. Since it is logically inconsistent to propose that the effect of R is dependent on τ but there is no effect of R, the main effect of R is retained in the model. The residual plot and normal probability plot for this model are shown in Figure A.1 (Appendix A.1). The residual plot against fitted k_1 showed that the variance increases as the fitted value increases. The normal probability plot indicates lack of normality. In an attempt to obtain a better model, two transformation methods appropriate to fix data with non-constant variance and non-normality are applied to k_1:

natural logarithm (Log) and square-root (Sqrt) transformations. Then, the regression analysis is applied to Log(k_1) and Sqrt(k_1) to fit the transformed k_1 to the main effects, interaction terms, quadratic terms and cubic terms. The residual plots and normal probability plots for Sqrt(k_1) and Log(k_1) are shown in Figure A.2 – A.3 (Appendix A.1), respectively. Comparing the normal probability plots for Sqrt(k_1) and Log(k_1), the residuals for Sqrt(k_1) still indicate lack of normality. Thus, natural logarithm transformation is chosen over square root. Recall that two interaction terms (M*τ and R*τ) were statistically significant in the regression model before any transformation was applied to k_1. However, when natural logarithm is applied to k_1, the only significant interaction term is R*τ. The final regression model selected for k_1 is:

$$\mathrm{Log}(k_1) = 1.8297 - 0.0326{*}M - 0.2628{*}R - 3.4394{*}\tau - 0.9927{*}R{*}\tau + 3.4555{*}\tau^2$$

This model is used to predict the value of k_1 for problem instances with J/M ratio equal to or larger than 1.7. The Analysis of Variance (ANOVA) tables and R^2 statistics for the regression model on k_1, Sqrt(k_1), and Log(k_1) are shown in Table A.1 – A.3 (Appendix A.1), respectively. The estimates of coefficient with standard errors for Log(k_1) model are shown in Table A.4 of Appendix A.1.

Recall that problem instances with J/M ratio less than 1.7 are generally not responsive to the changes in k_1. Thus, any value of k_1 from 0.2 to 8.0 is basically good for these problem instances. However, a small number of problem instances in this category have recommended k_1 values that are smaller than 1.5. Therefore, as k_1 values smaller than 1.5 are good for all problem instances with J/M ratio less than 1.7, it is reasonable to recommend a value of k_1 that lies within the range from 0.2 to 1.5 to these problem instances. For the experiments described in chapters 6 and 7, the values of k_1 used is fixed to 1.0 for problem instances with J/M ratio less than 1.7.

The analysis procedure for k_2 is similar to k_1. In the initial data exploration, the recommended values for k_2 were found to form two clusters: one cluster is around the smaller values (i.e. 0 to 2.0) and the other is around 3.1 to 3.3 with a concentration at 3.1. A similar situation was encountered in the initial data exploration for k_1. This leads to checking the data points that are concentrated at 3.1. The investigation is carried back to the range of values used to identify the recommended k_2 as a midpoint. Recall from the testing procedure, the range of k_2 is selected by considering the entire range of k_2 (0.2, 0.4, 0.6, ...6.0) with $\bar{k}_1$ that resulted in total weighted tardiness that is less than or equal to MTWT(1+β). It appeared that the range of k_2 that resulted in a midpoint of 3.1 is between 0.2 and 6.0. The implication of the entire range being selected is that each value from 0.2 to 6.0 yields the same TWT. This implies that problem instances with recommended k_2 of 3.1 are not responsive to the value of k_2 used in ATC rule. It means that their total weighted tardiness remains the same regardless of the value of k_2 being used. Thus, there is no need to find the function that relates k_2 to the values of the factors (J, M, τ, R) that correspond to these problem instances. The problem instances that obtained recommended k_2 of 3.1 have a ratio of J/M that is less than 1.7 or larger than 7.3. Thus, only problem instances with J/M ratio between 1.7 and 7.3 are considered further in the multiple linear regression analysis. The attempt is to fit k_2 to all main factors (J, M, τ, R), the interactions, quadratic term and cubic term of main factors. The effects that appeared to be statistically significant for $\alpha = 0.05$ are J, M, τ, J*M, and J*τ. The normal probability plot and residual plot for this model is shown in Figure A.4 (Appendix A.2). As the normal probability plot indicates that the distribution of the residuals is non-normal, square-root and natural logarithm transformation are applied to k_2. The normal probability plots and residual plots for the regression model on Log(k_2) and Sqrt(k_2) are shown in Figure A.5 – A.6 (Appendix A.2), respectively. The residual plot and normal probability plot for Log(k_2) are not much different from the plots for Sqrt(k_2). The main effect and interaction terms that appear to be significant are exactly the same for both models. However, the R^2 statistic, which is the percentage of variation explained by the model, for Sqrt(k_2) is greater than Log(k_2). Therefore, the square root transformation is selected over the natural logarithm. The final regression model selected for k_2 is:

$$\text{Sqrt}(k_2) = 2.2707 - 0.0174*J - 0.0912*M + 0.5022*\tau + 0.0017*J*M - 0.0193*J*\tau$$

The ANOVA tables and R^2 statistics for the regression models on k_2, Log(k_2), and Sqrt(k_2) are shown in Table A.5 – A.7 (Appendix A.2), respectively. The estimates of coefficients and standard errors for Sqrt(k_2) model are shown in Table A.8 (Appendix A.2). This model is used to predict the value of k_2 for problem instances that have J/M ratio lies between 1.7 to 7.3.

As most problem instances with J/M ratio less than 1.7 and larger than 7.3 are not responsive to the values of k_2 used, any value from 0.2 to 6.0 is basically good for these problem instances. However, a small number of problem instances in this category have recommended k_2 values that are smaller than 1.5. Therefore, as k_2 values smaller than 1.5 are good for all problem instances with J/M ratio less than 1.7 and larger than 7.3, it is reasonable to recommend a value of k_2 that lies within the range from 0.2 to 1.5 to these problem instances. For the experiments described in chapters 6 and 7, the values of k_2 used is fixed to 1.0 for problem instances with J/M ratio less than 1.7 and larger than 7.3.

For a particular problem instance, the appropriate values of k_1 and k_2 are estimated as follows:

- For k_1, calculate Log(k_1) using the selected regression model and take the exponent of the calculated result.
- For k_2, calculate Sqrt(k_2) using the selected regression model and take the square-term of the calculated result.

It is important to apply the functions (regression models) only to problem instances that have values of factors within the range used in the experiment, i.e. $5 \leq J \leq 70$, $3 \leq M \leq 20$, $0.2 \leq \tau \leq 0.8$, and $0.2 \leq R \leq 1.0$. To estimate k_1 and k_2 for a wider range of values of J, M, τ, and R, another experiment that incorporates those values has to be conducted.

Once the look-ahead parameters are evaluated for a particular problem instance, the priority function is ready to be applied. The split portions of a job are scheduled using the same mechanism as that in EDDsp or LFJ/LFM. It means that the necessary adjustment to the completion time is made on the split portions of a job to assure that the initial solution is feasible. The scheduling steps that use the modified ATC rule can be documented as:

1. Initially, set $t_i = 0$ and $mt_i = a_i \; \forall$ i. Include all jobs into NS.
2. Select the machine/unit (i) that has the minimum mt_i. If there are more than one machine with minimum mt_i, break ties by choosing the machine with the smallest machine index. Let the selected machine be i*. Set $t_i = mt_{i*}$.
3. Let SJ = the set of jobs that can be processed on i*, SJ $\subset$ NS. If SJ = $\emptyset$, exclude machine i* from future consideration and go to step 7. Calculate the priority index for each job in SJ using the priority function.

4. Identify the job with the highest priority index (PI). If the highest PI belongs to a non-split job, select that job. If the highest PI belongs to a split job, select the split portion of the job that has the highest processing time on machine i*. If two or more jobs tie for the highest PI and split jobs are among them, the priority is given to split jobs. If there are no split jobs among the ties, choose the job with the smallest job index.
5. Set j* = the selected job: ST(j*,i*) = max [t, r_{j*}], CT(j*,i*) = ST(j*,i*) + p_{i*j*} and mt_{i*} = CT(j*,i*). If j* is a non-split job, go to step 6. If j* is a split job, let the split portion of j* be j'*. Let SM be the set of machines that can process j'*. Calculate the starting and completion times of j'* on each machine included in SM: ST(j'*,k) = max [mt_k, $r_{j'*}$], CT(j'*,k) = ST(j'*,k) + $p_{kj'*}$, $k \in$ SM. Select the machine that can complete j'* earliest and call this machine i'*. Let CT(j'*,i'*) and ST(j'*,i'*) be the completion and start time of j'* on i'*, respectively. Check the difference between CT(j*,i*) and CT(j'*,i'*):
   a. If CT(j'*,i'*) - CT(j*,i*) > q_{j1j2}, set CT(j*,i*) = CT(j'*,i'*) - q_{j1j2}, ST(j*,i*) = CT(j*,i*) – p_{i*j*}. Set mt_{i*} = CT(j*,i*) and $mt_{i'*}$ = CT(j'*,i'*). Eliminate j* and j'* from NS. Go to step 7.
   b. If CT(j*,i*) - CT(j'*,i'*) > q_{j1j2}, set CT(j'*,i'*) = CT(j*,i*) - q_{j1j2}, ST(j'*,i'*) = CT(j'*,i'*) - $p_{i'*j'*}$. Set $mt_{i'*}$ = CT(j'*,i'*). Eliminate j* and j'* from NS. Go to step 7.
   c. If |CT(j*,i*) - CT(j'*,i'*)| ≤ q_{j1j2}, set $mt_{i'*}$ = CT(j'*,i'*). Eliminate j* and j'* from NS. Go to step 7.
6. Eliminate j* from NS.
7. If NS ≠ ∅, 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 at 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 moves 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 machine or two different machines. An insert move is a move that inserts a job to any machine except the one that it currently occupies. A swap move allows two jobs from the same or different machines to exchange positions. An insert move allows a job to move from one machine to another. The structure of solutions produced by swap moves is always the same as the structure of its parent solution (seed). In other words, swap moves do not change the total number of jobs that are assigned to each machine. On the contrary, insert moves always produce solutions that change the total number of jobs assigned to a machine. The swap move and insert move are described separately in the following two subsections.

5.4.1. Swap Move

Let JA_1 and JB_1 be the jobs considered for swap. JA_1 and JB_1 are currently scheduled on machine Ma and Mb, respectively. Let [...JA_0, JA_1, JA_2, JA_3...] be the partial sequence of jobs assigned to Ma and [...JB_0, JB_1, JB_2, JB_3...] be the partial sequence of jobs assigned to Mb. JA_1 and JB_1 are allowed to exchange positions if *all* of the following conditions are satisfied:

1. JA_1 can be processed on Mb and JB_1 can be processed on Ma.
2. $r_{JA_1} < CT(JB_1,Mb)$ and $r_{JB_1} < CT(JA_1,Ma)$.
3. If JA_1 is a split-job, the split portion of JA_1 is not scheduled on Mb. If JB_1 is a split-job, the split portion of JB_1 is not scheduled on Ma.

The third condition is used to avoid generating an infeasible solution. Scheduling two split portions of a job on the same machine would very unlikely satisfy the JIT constraints. The only exception will be when q_{j1j2} is large and/or the difference between the processing times of the split portions of a job is relatively large. However, it is unreasonable to use a large q_{j1j2} as the sole purpose of using it is to impose the JIT requirement on the completion times of the split portions. It is also unreasonable to have a large difference in processing times between two split portions. One might as well treat the job as a whole, not splitting it into two in the first place.

If JA_1 and JB_1 satisfied all three conditions, proceed with swapping JA_1 and JB_1. The start time and completion time of JA_1 and JB_1 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. $ST_r(JA_1,Mb)$ is set to be equal to the completion time of the job that precedes JB_1, i.e. $CT(JB_0,Mb)$, or r_{JA_1}, whichever is larger. $ST_r(JB_1,Ma)$ is set to be equal to the completion time of the job that precedes JA_1, i.e. $CT(JA_0,Ma)$, or r_{JB_1}, whichever is larger. If JB_1 is the first job in the sequence on Mb, then $ST_r(JA_1,Mb)$ is set to be equal to the max $[r_{JA_1}, a_{Mb}]$. On the other hand, if JA_1 is the first job in the sequence on Ma, set $ST_r(JB_1,Ma)$ to the max $[r_{JB_1}, a_{Ma}]$. $CT_r(JA_1,Mb)$ and $CT_r(JB_1,Ma)$ are then set to be equal to $ST_r(JA_1,Mb) + p_{MbJA_1}$ and $ST_r(JB_1,Ma) + p_{MaJB_1}$, respectively.

Once the swap move is applied, the start time and completion times of the jobs following JB_1 on Ma (i.e. JA_2, JA_3...) and JA_1 on Mb (i.e. JB_2, JB_3...) have to be revised accordingly. This is accomplished by setting $ST_r(JA_2,Ma) = \max\,[CT_r(JB_1,Ma), r_{JA_2}]$ and $CT_r(JA_2,Ma) = ST_r(JA_2,Ma) + p_{MaJA_2}$; $ST_r(JA_3,Ma) = \max\,[CT_r(JA_2,Ma), r_{JA_3}]$ and $CT_r(JA_3,Ma) = ST_r(JA_3,Ma) + p_{MaJA_3}$; and so on. The revision is performed in the same manner for all jobs following JA_1 on Mb.

Split jobs may be involved in the swapping process in several ways. One or both swapped jobs may be split jobs. Split jobs may be included in the sequence of jobs that follow the swapped jobs. In any case, these situations require further investigation to ensure that the split jobs satisfy the JIT requirement. If the JIT constraint is violated between the split portions of a job, the start and completion times of one of the split portion have to be adjusted to attain a feasible schedule. This adjustment is categorized into three levels as follows:

1. First-level adjustment

   The first-level adjustment is applicable if JA_1 and/or JB_1 is a split job and the JIT constraint for these split jobs is violated. The completion time of JA_1 must be compared with the completion time of its split portion. Let JA_1' be the split portion of JA_1 and Mc is the machine to which JA_1' is assigned. If the absolute difference between $CT_r(JA_1,Mb)$ and $CT(JA_1',Mc)$ is larger than the maximum allowance, q_{j1j2}, ST and CT of JA_1' have to be revised so that the JIT constraint for split jobs is not violated. The ST and CT of JA_1' are revised so that the difference between $CT_r(JA_1,Mb)$ and $CT_r(JA_1',Mc)$ is not any larger than q_{j1j2}. As the job swapping is initiated by JA_1, a sequence change for JA_1' on Mc would be considered to attain feasibility. If any sequence change has to be made, the job, whose position on Mc is taken by JA_1' should be sequenced right after JA_1'. If JB_1 is a split job, let JB_1' be the split portion of JB_1 and Md be the machine to which JB_1' is assigned. The absolute difference between $CT_r(JB_1,Ma)$ and $CT(JB_1',Md)$ must be within the maximum allowance, q_{j1j2}. If the JIT requirement is violated, the same process of revision applied to JA_1' should be applied JB_1'.

2. Second-level adjustment

   The second-level adjustment is applicable if one or more split jobs are sequenced following JA_1 on Mb or JB_1 on Ma, and the JIT constraint for these split jobs is violated. It is also applicable if JA_1 or JB_1 is a split job, and the split jobs that are sequenced following JA_1' on Mc or JB_1' on Md violate the JIT constraint. Identify any violation of the JIT constraint in the completion times of these split jobs. If any violation is detected, sequence changes are not allowed to attain feasibility since the associated split job is not the split portion of the job that initiated swapping. Such a split job, if exists, would be moved either forward or backward to attain feasibility without changing the sequence on the machine.

3. Third-level adjustment

   The third-level adjustment is applicable to the split jobs that are not sequenced on Ma, Mb, Mc and Md. Let Mx be one of the machines on which these split jobs are scheduled. Identify any violation of the JIT constraint in the completion times of the split jobs scheduled on Mx. If any violation is detected, sequence changes on Mx are not allowed to attain feasibility since the associated split job is not the split portion of the job that initiated swapping. Let JX be the split job on Mx that violates the JIT constraint. The start and completion times of JX are revised only if JX falls into one of the following criteria:

a. JX is the last job sequenced on Mx, or
b. All jobs that are sequenced following JX on Mx are non-split jobs, or
c. JX is followed by only one split job. Let JY be that split job, JY' be the split portion of JY, and My be the machine on which JY' is scheduled. JY' should be the last job sequenced on My, or all jobs that are sequenced following JY' on My are non-split jobs.

In the process of revising and adjusting the start time and completion time of the split portions of a job to attain feasibility, the start time of a split portion may need to be delayed, which would cause machine idleness. Intuitively, a problem instance that consists of two or more pairs of split jobs could get continuous revision on the start time and completion time of several jobs. This may cause unending revision on the schedule. For this reason, the criteria in the third-level adjustment are added in order to prevent the schedule from being repeatedly adjusted.

5.4.2. Insert Move

Let JA_1 be the job considered to be inserted into the position that is currently occupied by JB_1 on machine Mb. JA_1 is currently scheduled on Ma. Let [...JA_0, JA_1, JA_2, JA_3...] be the partial sequence of the jobs assigned to Ma and [...JB_0, JB_1, JB_2, JB_3...] be the partial sequence of the jobs assigned to Mb. JA_1 can take over the position occupied by JB_1 on Mb if *all* of the following conditions are satisfied:

1. Mb is capable of processing JA_1.
2. $r_{JA_1} < CT(JB_1, Mb)$.
3. If JA_1 is a split job, the split portion of JA_1 is not scheduled on Mb.

After inserting JA_1 on Mb, the partial sequence of jobs on Ma and Mb will appear as follows: Ma: [...JA_0, JA_2, JA_3...] and Mb: [...JB_0, JA_1, JB_1, JB_2, JB_3...]. Before determining the start time of JA_1, a check has to done if JA_1 is preceded by any split jobs and if the JIT requirement on the split portions of these jobs is violated. If the JIT requirement is violated, the start and completion times of these split portions would be revised to attain feasibility without changing job sequences. $ST_r(JA_1,Mb)$ is then set equal to max $[CT_r(JB_0,Mb), r_{JA_1}]$. If there is no job that precedes JB_1 (JB_1 is the first job in the sequence on Mb), set $ST_r(JA_1,Mb)$ to be the max $[r_{JA_1}, a_{Mb}]$. $CT_r(JA_1,Mb)$ is equal to $ST_r(JA_1,Mb)$ plus p_{MbJA_1}. The start time and completion time of the jobs following JA_1 on Mb (i.e. JB_1, JB_2, JB_3...) and the remaining jobs on Ma (i.e. JA_2, JA_3...) have to be revised accordingly. On Mb, set $ST_r(JB_1,Mb) = \max [CT_r(JA_1,Mb), r_{JB_1}]$ and $CT_r(JB_1,Mb) = ST_r(JB_1,Mb) + p_{MbJB_1}$; set $ST_r(JB_2,Mb) = \max [CT_r(JB_1,Mb), r_{JB_2}]$ and $CT_r(JB_2,Mb) = ST_r(JB_2,Mb) + p_{MbJB_2}$; and so on. On Ma, set $ST_r(JA_2,Ma) = \max [CT(JA_0,Ma), r_{JA_2}]$ and $CT_r(JA_2,Ma) = ST_r(JA_2,Ma) + p_{MaJA_2}$; set $ST_r(JA_3,Ma) = \max [CT_r(JA_2,Ma), r_{JA_3}]$ and $CT_r(JA_3,Ma) = ST_r(JA_3,Ma) + p_{MaJA_3}$; and so on.

Further investigation is required to ensure that the split jobs satisfy the JIT requirement. If the JIT constraint is violated between the split portions of a job, the start and completion times of one of the split portion have to be adjusted to attain feasible schedule. This adjustment is categorized into three levels as follows:

1. First-level adjustment

   The first-level adjustment is applicable if JA_1 is a split job and the JIT constraint for JA_1 and its split portion is violated. The completion time of JA_1 must be compared with the completion time of its split portion. Let JA_1' be the split portion of JA_1 and Mc is the machine to which JA_1' is assigned. If the absolute difference between $CT_r(JA_1,Mb)$ and $CT(JA_1',Mc)$ is larger than the maximum allowance, q_{j1j2}, ST and CT of JA_1' have to be revised so that the JIT constraint is not violated. The ST and CT of JA_1' are revised so that the difference between $CT_r(JA_1,Mb)$ and $CT_r(JA_1',Mc)$ is not any larger than q_{j1j2}. As the job insertion is initiated by JA_1, a sequence change for JA_1' on Mc would be considered to attain feasibility. If any sequence change has to be made, the job, whose position on Mc is taken by JA_1' should be sequenced right after JA_1'.

2. Second-level adjustment

   The second-level adjustment is applicable if one or more split jobs are sequenced following JA_1 on Mb or JA_0 on Ma, and the JIT constraint for these split jobs is violated. It is also applicable if JA_1 is a split job, and the split jobs that are sequenced following JA_1' on Mc violate the JIT constraint. Identify any violation of the JIT constraint in the completion times of these split jobs. If any violation is detected, sequence changes are not allowed to attain feasibility since the associated split job is not the split portion of the job that initiated the insert move. Such a split job, if exists, would be moved either forward or backward to attain feasibility without changing the sequence on the machine.

3. Third-level adjustment

   The third-level adjustment is applicable to the split jobs that are not sequenced on Ma, Mb, and Mc. Let Mx be one of the machines on which these split jobs are scheduled. Identify any violation of the JIT constraint in the completion times of the split jobs scheduled on Mx. If any violation is detected, sequence changes on Mx are not allowed to attain feasibility since the associated split job is not the split portion of the job that initiated the insert move. Let JX be the split job on Mx that violates the JIT constraint. The start and completion times of JX are revised only if JX falls into one of the following criteria:

   a. JX is the last job sequenced on Mx, or
   b. All jobs that are sequenced following JX on Mx are non-split jobs, or
   c. JX is followed by only one split job. Let JY be that split job, JY' be the split portion of JY, and My be the machine on which JY' is scheduled. JY' should be the last job sequenced on My, or all jobs that are sequenced following JY' on My are non-split jobs.

The criteria in the third-level adjustment are included for the same reason as in the swap move, i.e. to prevent unending revision on the schedule.

5.5. Steps of Tabu Search

The steps based on tabu-search mechanism are documented as follows:

Step 1: Using the initial solution as seed, apply swap moves and insert moves to obtain a set of neighborhood solutions. The swap moves are attempted on any possible combinations of two jobs. A problem instance with n jobs has $\binom{n}{2} = \frac{n!}{(n-2)!*2!}$ possible combinations of swap moves. A swap move is applied to a pair of jobs if they satisfy the conditions listed in Section 5.4.1. In applying insert moves, the attempt is to insert every job to different positions in all machines except the machine that the job is currently occupying. The total number of positions on a machine to insert a job to is equal to all occupied positions plus one last unoccupied position. For example, if k positions on a machine are initially occupied, then a job is inserted to k + 1 positions on the same machine. The conditions for insert move listed on Section 5.4.2 must be satisfied before the move is applied. The results of both swap and insert moves are a set of solutions considered as the neighborhoods of the initial solution.

Step 2: Evaluate the total weighted tardiness (TWT) of every solution in the neighborhood. Schedules that violate maximum permissible constraint on split jobs are given a penalty. This penalty is introduced as a big number added to the TWT value, which in turn reflects the infeasibility built into the solution.

Step 3: Select the solution that yields the best (minimum) TWT value. If there is more than one best solution, choose the first-best solution. Apply the move that results in the selected best solution to the initial solution. The following parameters used for tabu search have to be updated:

(1) Tabu list: This list consists of the most recent moves. The move that results in the selected best solution must be recorded. If it is a swap move, record in the tabu list the pair of jobs being exchanged. The pairs of jobs that appear in the tabu list indicate that these pairs have been swapped before at some previous iterations. These pairs of jobs are not allowed to exchange positions in the next iteration unless an aspiration criterion is satisfied. If the best solution is the result of an insert move, record in the tabu list the job index along with the position and machine occupied by the job before the move was

applied. The job, position and machine that appear in the tabu list indicate that this job is not allowed to be inserted back to this position and machine in the next iteration unless an aspiration criterion is satisfied.

The length of time a move remains tabu depends on the size of tabu list. For example, if the size of tabu list is equal to 4, then a move will stay tabu for four iterations. The entries in the tabu list are first-in-first-out (FIFO). When the tabu list is stored up to its size, the oldest entry is removed before the new one is entered.

Since tabu list stores the 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. The total number of possible moves of a solution would increase as the number of jobs and machines increases, and decrease as the number split jobs increase. Therefore, the size of tabu list is dependent on the total number of jobs, machines and total pairs of split jobs. Two different types of tabu list size are studied in this research: fixed tabu list size and variable tabu list size. An initial experimentation was conducted to determine appropriate tabu list size for different problem instances. Based on the result of the experiment, the following formulae are developed:

- For fixed tabu list size, use the following formula:
  The fixed size of tabu list = $\mathrm{INT}(N*\sqrt{M}/(3*\sqrt{SP}))$
- For variable tabu list size, use the following formulae:
  The initial size of tabu list = $\mathrm{INT}(N*\sqrt{M}/(3*\sqrt{SP}))$
  The decreased size of tabu list = $\mathrm{INT}(N*\sqrt{M}/(4*\sqrt{SP}))$
  The increased size of tabu list = $\mathrm{INT}(N*\sqrt{M}/(2.5*\sqrt{SP}))$

where N is the total number of jobs, M is the total number of machines, and SP is the total number of pairs of split jobs.

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

(2) <u>Aspiration Level (AL)</u>: Aspiration criterion is the condition a tabu move has to satisfy in order to be released from its tabu restriction. At the beginning of the search process, Aspiration Level (AL) is set equal to the TWT of the initial solution. At every

iteration, if the TWT of the selected best solution is less than AL, it is updated to be equal to the TWT of the selected best solution. If a tabu move results in TWT that is better than AL, the move is released from tabu restriction and its corresponding schedule is included in the set of solutions considered for selection.

(3) Candidate List (CL) and Index List (IL): Candidate List consists of the best solution selected at each iteration. Index List consists of all local optima evaluated during the search process. The initial solution (S_0) is considered as the first local optimum, therefore it is admitted to the IL as well as the CL. As implied in Step 2, the total weighted tardiness (TWT) of a solution is used as the performance measure. Suppose that the best solution obtained by perturbing S_0 is S_1. S_1 is admitted to CL. If TWT of S_1 < TWT of S_0, S_1 will receive a star (*), which indicates that it has the potential to become a local optimum. Suppose that the iteration after S_1 results in S_2. If TWT of $S_1 \leq$ TWT of S_2, then S_1 will receive another star. Otherwise, S_2 will receive a star. A solution that receives double star (**) implies that it is the next local optimum and is admitted to the IL. At every iteration, before a solution is admitted to the CL, it has to be checked against all entries in the CL. If the solution already exists in the CL, another best solution has to be chosen instead.

(4) Number of iterations without improvement (IT): Initially, the number of iterations without improvement is set to zero. If there is no improvement in the total weighted tardiness value (i.e. the current TWT is equal to or larger than previous TWT), increase the number of iterations without improvement by one. Once an improvement in the TWT is made, reset the number of iterations without improvement to zero.

(5) Long-term memory (LTM) matrix: The long-term memory matrix is a frequency matrix that keeps track of the number of times a job is processed on a particular machine. It is used when the algorithm employs the long-term memory function of tabu search. The size of the LTM matrix is equal to the number of jobs times the number of machines. Initially, all entries or cells in the LTM matrix are set equal to zero. Exception is given to the cells that correspond to the jobs that cannot be processed on certain machines; these cells will remain empty throughout the search. The LTM matrix is updated at every iteration. Every time an iteration is made, each cell that corresponds to the machine on which a job is processed, is increased by one. The LTM matrix provides information

about which machine is the most or the least frequently used by a job. This information is used to determine the restarting point for diversifying the search process.

__Step 4__: The stopping criteria used to terminate the search are the maximum number of iterations without improvement (ITmax) and maximum entries into the IL (ILmax). Both criteria are dependent on the size of the problem instance, which is proportional to the total number of jobs. ITmax increases as the number of jobs increases, so does ILmax. Based on preliminary experimentation, ITmax and ILmax are developed as:

$$ITmax = INT(N/6.25)$$

$$ILmax = INT(N/4)$$

where N is the total number of jobs.

The stopping criteria for fixed and variable size of tabu list are as follows:

- For the fixed tabu list, the search is terminated if ITmax or ILmax is reached, whichever is activated first.
- For the variable tabu list, ILmax is used in conjunction with the following steps:
  (i) If there is no improvement in the last $\lceil ITmax/3 \rceil$ 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 $\lceil ITmax/3 \rceil$ 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 $\lceil ITmax/3 \rceil$ iterations with the increased size of tabu list, terminate the intensification search.

If neither of the stopping criteria is met, repeat Step 1 to Step 3 on the selected best solution instead of the initial solution. The process is repeated until a stopping criterion is met.

__Step 5__: The intensification and diversification strategy of tabu search can be applied by using the information provided by the LTM matrix. Two different approaches are taken to diversify the search: long term memory based on maximum frequency (LTM-max) and long term memory based on minimum frequency (LTM-min). The first approach directs the search to restart from the regions that are considered 'good' during the previous search. The second approach directs the search to restart from the regions that were least or never explored before. The guidelines for the use of LTM matrix are developed as follows:

- For LTM-max, select the job-machine pair with maximum frequency in the matrix, and fix the job to the respective machine throughout the search until the next restart is invoked. If there is a tie in maximum frequency, use row-wise first-best strategy. Care should be taken in identifying the cell with maximum frequency. If there is one or more jobs that can be processed on only one machine, the tally in the cells corresponding to these jobs processed on the machine would be the maximum. It is meaningless to fix these jobs to the machine since without doing so the jobs will not be processed on other machine throughout the course of the search process. Thus, the job-machine pair with maximum frequency should be selected from among the jobs that can be processed on two or more machines.
- For LTM-min, select the job-machine pair with minimum frequency in the matrix, and fix the job to the respective machine throughout the search until the next restart is invoked. Similar to 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 machine, which it is supposed to fix on, is referred to as 'fixed machine'. The schedule used for restart is generated from the initial solution. The restart solution will be similar to the initial solution if 'fixed job' is already assigned to 'fixed machine' in the initial solution. The difference between the initial solution and the restart solution is the 'fixed job' will not be removed from the 'fixed machine' until another restart is invoked. If 'fixed job' is not assigned to 'fixed machine' in the initial solution, the restart solution is generated by applying insert move to the initial solution, i.e. the 'fixed job' is inserted to the first position of the 'fixed machine'. The insertion to the *first* position of 'fixed machine' is preferred to incorporate a situation such as no jobs are processed on the 'fixed machine' and thus, 'fixed job' will become the first job as well as the only job processed on it. If 'fixed job' is a split job and its split portion is assigned to 'fixed machine' in the initial solution, then inserting 'fixed job' to 'fixed machine' will very likely result in an infeasible solution. Thus, to overcome this problem, a swap move is used instead of an insert move to exchange the 'fixed job' with its split portion.

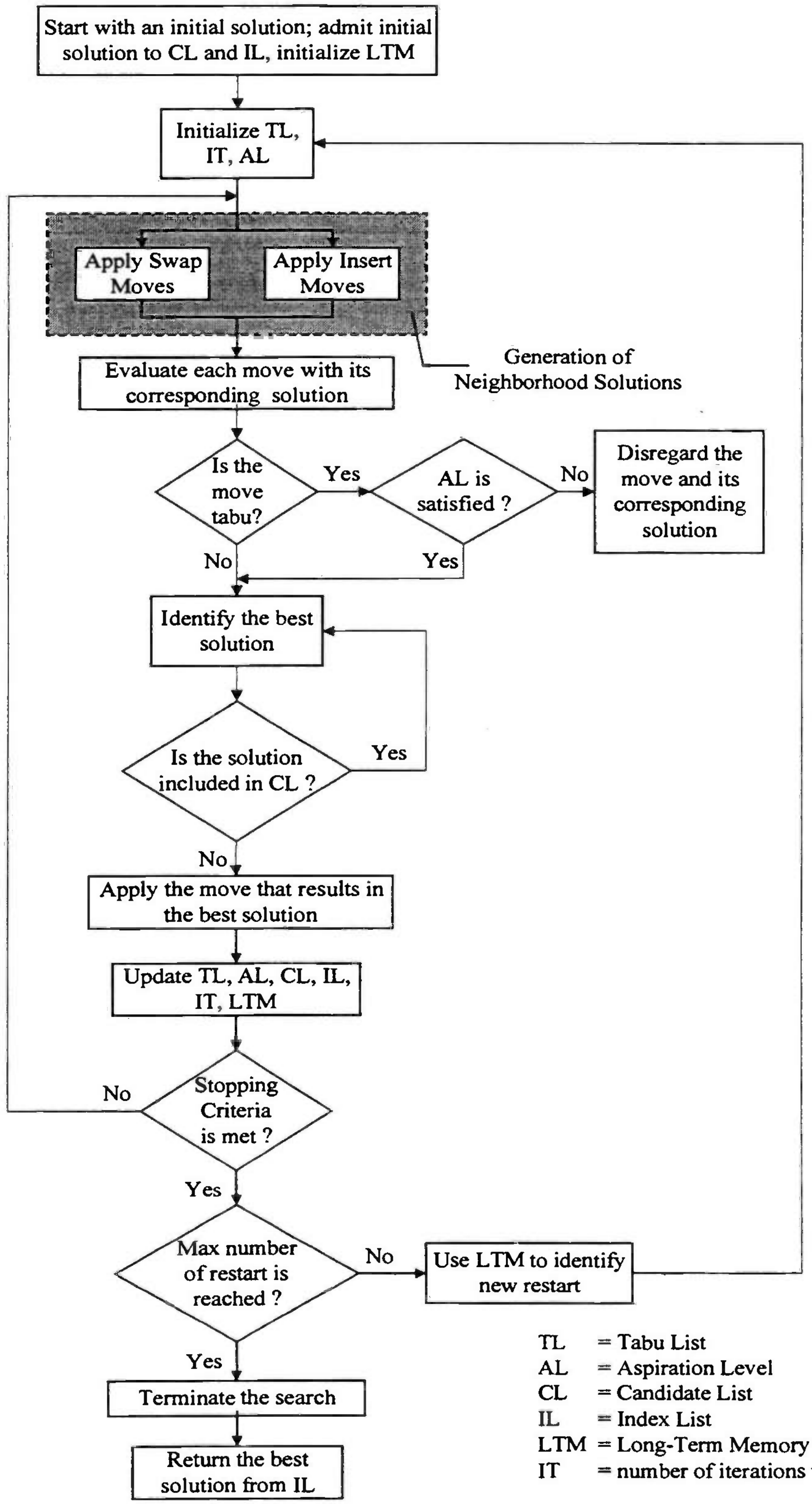
The tabu list, AL and IT must be re-initialized at the beginning of each restart. Using the restart solution as a starting point, repeat Step 1 to Step 4. The total number of restart used in this research is assumed to be 2. Two restarts have been used by Logendran and Sonthinen (1997), and Karim (1999).

Step 6: Utilizing only the short-term memory function of tabu search, the entire search should be terminated at the end of Step 4. For long-term memory function, the entire search is terminated when the total number of restarts is reached. For both memory functions, the optimal/near-optimal solution is the solution with the minimum TWT selected from the Index List.

The steps of the tabu-search based heuristics are summarized in the flow chart shown in Figure 5.1. The programming for the entire algorithmic steps were written in commands for MATLAB version 4.2 (The MathWorks, 1984-1994). The programs are written in the form of script files and function files, which are executable from MATLAB.

5.6. Application of Heuristic Algorithm to Example Problem

An example problem is presented to illustrate the application of the heuristic. The example problem, shown in Table 5.2, involves nine jobs and four machines. This example was carefully constructed to represent a situation that is typically found in the industry. Three different types of machines are considered. These include 1 unit of the most capable machine (M1), two units of the least capable machine (M31 and M32), and one unit of machine with medium capability (M2). M31 and M32 are identical machines. The capability of each machine is indicated by the length of time each machine takes to process a job. Take J2 as an example: M1, as the most capable machine, takes relatively shorter time (4 units of time) than M2 (8 units) and M31/M32 (9 units). Jobs that cannot be processed on certain machines are assumed to have infinite processing time such as J5 on M1. Infinite processing time is indicated by infinity sign (∞) in Table 5.2.



TL = Tabu List
AL = Aspiration Level
CL = Candidate List
IL = Index List
LTM = Long-Term Memory matrix
IT = number of iterations without improvement

Figure 5.1 Flow chart of tabu-search based heuristic

There are two pairs of split jobs in this example: J41-J42 and J71-J72. The size of the split portions of J4 and J7 is not equal. J4 when split, results in processing time of 7 and 8 on M1, and 9 and 11 on M2. This shows a general case where an unequal split results in unequal processing times that are close. If the size of the split portions is equal, the split will result in equal processing time. The challenge for a scheduler is to split the job more or less equally to obtain unequal processing times that are close. The maximum allowable limit for the difference in completion time between two split portions, q_{j1j2}, is assumed equal to 1 time unit. The split portions of a job have to be scheduled in such a way that they meet constraints (7) and (8) mentioned in Section 4.4. Each machine is made available at different times: M1 is available at $t = 0$, M2 and M31 at $t = 2$, and M32 at $t = 5$. Each job has a different weight, release time and due date. Split jobs that came from the same 'batch' have the same weight, release time and due date. The four methods described in Section 5.3 are applied to this example problem to obtain initial solutions.

Table 5.2 Example problem with 9 jobs and 4 machines

	Machine				**Job Weight**	**Job Release Time**	**Job Due Date**
	M1	M2	M31	M32			
Machine Availability	0	2	2	5			
Job	**Job Processing Time on Machine**						
J1	10	∞	∞	∞	1	1	15
J2	4	8	9	9	2	4	12
J3	∞	5	8	8	2	3	7
J41	7	9	∞	∞	3	4	10
J42	8	11	∞	∞	3	4	10
J5	∞	4	6	6	2	9	18
J6	8	10	∞	∞	1	8	20
J71	∞	6	9	9	2	5	11
J72	∞	7	11	11	2	5	11

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

- At t = 0, mt_{M1} = 0, mt_{M2} = 2, mt_{M31} = 2, mt_{M32} = 5. NS = [J1,J2,J3,J41,J42,J5,J6,J71,J72].
- The machine with minimum mt_i is M1, t = 0. No job is released at t. Since J1 can be processed on M1 and has minimum r_j, J1 is selected to be processed on M1. ST(J1,M1) = max [0,1] = 1, CT(J1,M1) = 1 + 10 = 11, mt_{M1} = 11. NS = [J2,J3,J41,J42,J5,J6,J71,J72].
- M2 and M31 have minimum mt_i. M2 is selected over M31 because M2 has smaller index. Setting t = 2, no unscheduled jobs are released earlier than or at t. Since J3 can be processed on M2 and has minimum r_j , J3 is assigned to M2. ST(J3,M2) = max [2,3] = 3, CT(J3,M2) = 3 + 5 = 8, mt_{M2} = 8. NS = [J2,J41,J42,J5,J6,J71,J72].
- The next machine with minimum mt_i is M31, t = 2. No unscheduled jobs are released earlier than or at t. Since J2 can be processed on M31 has minimum r_j, J2 is selected to be processed on M31. ST(J2,M31) = max [2,4] = 4, CT(J2,M31) = 4 + 9 = 13, mt_{M31} = 13. NS = [J41,J42,J5,J6,J71,J72].
- The next machine with minimum mt_i is M32 and t = 5. SJ = [J71,J72]. Note that J71 and J72 are split jobs of the same batch. Since $p_{M32J72} > p_{M32J71}$, J72 is chosen to be assigned to M32. ST(J72,M32) = 5, CT(J72,M32) = 16, mt_{M32} = 16. NS = [J41,J42,J5,J6,J71].
- mt_i = [11,8,13,16]. M2 has the minimum mt_i and t = 8. SJ = [J41,J42,J6,J71]. J71 is selected from SJ list because the split portion of J71, i.e. J72 was scheduled. ST(J71,M2) = 8 and CT(J71,M2) = 14. As CT(J72,M32) – CT(J71,M2) = 2 > 1 (i.e. q_{j1j2}), the start time of J71 is delayed. Thus, CT(J71,M2) = 16 –1 = 15, ST(J71,M2) = 15 – 6 = 9, mt_{M2} = 15. NS = [J41,J42,J5,J6].
- mt_i = [11,15,13,16]. M1 has the minimum mt_i and t = 11. SJ = [J41,J42,J6]. The jobs in SJ list that have the EDD are J41 and J42. Since $p_{M1J42} > p_{M1J41}$, J42 is selected over J41 and assigned to M1. ST(J42,M1) = 11, CT(J42,M1) = 19, mt_{M1} = 19. NS = [J41,J5,J6].

- mt_i = [19,15,13,16]. M31 has the minimum mt_i, t = 13, and SJ = [J5]. Assign J5 to M31 with ST(J5,M31) = 13, CT(J5,M31) = 19, mt_{M31} = 19. NS = [J41,J6].
- mt_i = [19,15,19,16]. M2 has the minimum mt_i, t = 15, and SJ = [J41,J6]. Since the split portion of J41 (i.e. J42) is scheduled previously, J41 is chosen to be scheduled on M2. ST(J41,M2) = 15 and CT(J41,M2) = 24. Although the JIT constraint for J41 and J42 is violated (i.e. CT(J41,M2) – CT(J42,M1) = 5 > 1), no attempt is made to achieve feasibility because J42 was scheduled before and its assignment is considered permanent. Thus, this initial solution is infeasible. Set mt_{M2} = 24 and NS = [J6].
- mt_i = [19,24,19,16]. The machine with minimum mt_i is M32. However, the remaining unscheduled job (i.e. J6) cannot be processed on M32. Thus, M32 is excluded from future consideration.
- Excluding M32, mt_i = [19,24,19]. The machines with minimum mt_i are M1 and M31. M1 is selected over M31 because M1 is the machine with smaller index. J6 is the only job left to be scheduled. J6 is assigned to M1 with ST(J6,M1) = 19, CT(J6,M1) = 27, and mt_{M1} = 27.

b. The following evaluations are obtained by applying the EDDsp method to the example problem:

- At t = 0, mt_{M1} = 0, mt_{M2} = 2, mt_{M31} = 2, mt_{M32} = 5. NS = [J1,J2,J3,J41,J42,J5,J6,J71,J72].
- The machine with minimum mt_i is M1, t = 0. No job is released at t. Since J1 can be processed on M1 and has minimum r_j, J1 is selected to be processed on M1. ST(J1,M1) = max [0,1] = 1, CT(J1,M1) = 1 + 10 = 11, mt_{M1} = 11. NS = [J2,J3,J41,J42,J5,J6,J71,J72].
- Both M2 and M31 have minimum mt_i. M2 is selected over M31 because M2 is the machine with smaller index. Setting t = 2, no jobs are released earlier than or at t. Since J3 can be processed on M2 and has minimum r_j, J3 is assigned to M2. ST(J3,M2) = max [2,3] = 3, CT(J3,M2) = 3 + 5 = 8, mt_{M2} = 8. NS = [J2,J41,J42,J5,J6,J71,J72].

- The next machine with minimum mt_i is M31, t = 2. No jobs are released earlier than or at t. Since J2 can be processed on M31 and has minimum r_j, J2 is selected to be processed on M31. ST(J2,M31) = max [2,4] = 4, CT(J2,M31) = 4 + 9 = 13, mt_{M31} = 13. NS = [J41,J42,J5,J6,J71,J72].
- The next machine with minimum mt_i is M32 and t = 5. SJ = [J71,J72]. Note that J71 and J72 are split jobs of the same batch. Since $p_{M32J72} > p_{M32J71}$, J72 is chosen to be assigned to M32. ST(J72,M32) = 5, CT(J72,M32) = 16 and mt_{M32} = 16. At this point, the attempt is to find a machine that J71 can be assigned to and can be completed at the earliest time. SM, the group of machines that can process J71, consists of M2, M31 and M32. The tentative start time and completion time of J71 on these machines are as follows:
  M2: ST = max [8,5] = 8 and CT = 8 + 6 = 14;
  M31: ST = max [13,5] = 13 and CT = 13 + 9 = 22;
  M32: ST = max [16,5] = 16 and CT = 16 + 9 = 25;
  J71 can be completed earliest on M2. Thus, J71 is assigned to M2. Since CT(J72,M32) – CT(J71,M2) = 2 > q_{j1j2}, delay the start time of J71 so that CT(J71,M2) = 15 and ST(J71,M2) = 9. mt_{M2} = 15, mt_{M32} = 16 and NS = [J41,J42,J5,J6].
- mt_i = [11,15,13,16]. The machine with minimum mt_i is M1, t = 11 and SJ = [J41,J42,J6]. Both J41 and J42 have the same EDD. As $p_{M1J42} > p_{M1J41}$, J42 is chosen over J41. ST(J42,M1) = 11, CT(J42,M1) = 19, and mt_{M1} = 19. At this point, the challenge is to find a machine on which J41 can be completed at the earliest time. SM = [M1,M2]. The tentative start time and completion time of J41 on these machines are as follows:
  M1: ST = max [19,4] = 19 and CT = 19 + 7 = 26;
  M2: ST = max [15,4] = 15 and CT = 15 + 9 = 24;
  J41 can be completed earliest on M2. Thus, J41 is assigned to M2. Since CT(J41,M2) – CT(J42,M1) = 24 – 19 = 5 > q_{j1j2}, the start time of J42 on M1 is delayed such that CT(J42,M1) = 23 and ST(J42,M1) = 15. mt_{M1} = 23, mt_{M2} = 24, and NS = [J5,J6].

- mt_i = [23,24,13,16]. The machine with min mt_i is M31, t = 13 and SJ = [J5]. J5 is assigned to M31 with ST(J5,M31) = 13 and CT(J5,M31) = 19. mt_{M31} = 19 and NS = [J6].
- mt_i = [23,24,19,16]. The last job to be scheduled is J6. It can only be processed on M1 or M2. Since M1 is released earlier than M2, J6 is assigned to M1 with ST(J6,M1) = 23 and CT(J6,M1) = 31. mt_{M1} = 31.

c. The following evaluations are obtained by applying the LFJ/LFM method to the example problem:
   - Set t = 0; mt_i = [0, 2, 2, 5]; NS = [J1,J2,J3,J41,J42,J5,J6,J71,J72].
   - Since no jobs are released at t = 0, t is advanced to 1 (i.e. min release time of all unscheduled jobs). Since J1 is the only job released at t = 1, J1 is selected and assigned to M1 as the only available machine. ST(J1,M1) = 1 and CT(J1,M1) = 11. mt_{M1} = 11, NS = [J2,J3,J41,J42,J5,J6,J71,J72], newt = min [11,2,2,5] = 2, t = max [1,2] = 2.
   - Since no jobs are released earlier than or at t = 2, set t = min [r_j] = 3. Since J3 is the only job released at t, J3 is selected. Both M2 and M31 are available and capable of processing J3. M2 is capable of processing a total of 8 out of 9 jobs, while M31 is capable of processing 5 jobs. Since M31 is less flexible than M2, M31 is selected. Assign J3 to M31 with ST(J3,M31) = 3 and CT(J3,M31) = 11. mt_{M31} = 11, NS = [J2,J41,J42,J5,J6,J71,J72], newt = min [11,2,11,5] = 2 and t = 3.
   - Since no jobs are released earlier than or at t = 3, set t = min [r_j] = 4. J2, J41, J42 are released at t = 4. The least flexible among these three jobs are J41 and J42. The least flexible machine that can process both J41 and J42 is M1. However, M1 is not available until t = 11. The only capable machine that is currently idle is M2. Since $p_{M2J42} > p_{M2J41}$, assign J42 to M2 with ST(J42,M2) = 4, CT(J42,M2) = 15 and mt_{M2} = 15. At this point, the attempt is to find a machine that can complete J41 earliest. SM = [M1,M2]. The tentative start time and completion time of J41 on these machines are as follows:
     M1: ST = max [11,4] = 11 and CT = 11 + 7 = 18;

M2: ST = max [15,4] = 15 and CT = 15 + 9 = 24;

As M1 can complete J41 earliest, J41 is assigned to M1. Since CT(J41,M1) – CT(J42,M2) = 3 > q_{j1j2}, the start time of J42 on M2 is delayed such that CT(J42,M2) = 17 and ST(J42,M2) = 6. mt_{M1} = 18, mt_{M2} = 17, NS = [J2,J5,J6,J71,J72]. Set newt = min [18,17,11,5] = 5, t = max [4,5] = 5.

- At t = 5, J2, J71 and J72 have been released. The least flexible among these jobs are J71 and J72. As M32 is the only available machine, it is selected. Since $p_{M32J72} > p_{M32J71}$, assign J72 to M32 with ST(J72,M32) = 5, CT(J72,M32) = 16 and mt_{M32} = 16. At this point, the attempt is to find a machine that can complete J71 earliest. SM = [M2,M31,M32]. The tentative start time and completion time of J71 on these machines are as follows:

  M2: ST = max [17,5] = 17 and CT = 17 + 6 = 23;

  M31: ST = max [11,5] = 11 and CT = 11 + 9 = 20;

  M32: ST = max [16,5] = 16 and CT = 16 + 9 = 25;

  As M31 can complete J71 earliest, J71 is assigned to M31. Since CT(J71,M31) – CT(J72,M32) = 4 > q_{j1j2}, the start time of J72 on M32 is delayed such that CT(J72,M32) = 19 and ST(J42,M32) = 8. mt_{M31} = 20, mt_{M32} = 19, NS = [J2,J5,J6]. Set newt = min [18,17,20,19] = 17, t = max [5,17] = 17.
- At t = 17, J2, J5 and J6 have been released. The least flexible among the released jobs is J6. The only available machine that is capable of processing J6 is M2. Thus, J6 is assigned to M2 with ST(J6,M2) = 17 and CT(J6,M2) = 27. mt_{M2} = 27, NS = [J2,J5], newt = min [18,27,20,19] = 18, t = max [17,18] = 18.
- At t = 18, J2 and J5 have been released. The least flexible among the released jobs is J5. All machines capable of processing J5 are not available at t = 18. Thus, newt and t are updated. From all machines that are capable of processing J5, newt = min [27,20,19] = 19 and t = 19.
- At t = 19, J2 and J5 have been released. The least flexible job is J5. The only available machine that is capable of processing J5 is M32. Thus, J5 is assigned to M32 with ST(J5,M32) = 19 and CT(J5,M32) = 25. mt_{M32} = 25, NS = [J2], newt = min [18,27,20,25] = 18, t = max [19,18] = 19.

- Finally, the only job left is J2. The available machine at t = 19 is M1. Thus, J2 is assigned to M1 with ST(J2,M1) = 19 and CT(J2,M1) = 23. mt_{M1} = 23, newt = min [23,27,20,25] = 20, t = max [19,20] = 20.

d. The following evaluations are obtained by applying the ATC method to the example problem:

Prior to applying the ATC method to the example problem, it is necessary to determine the appropriate look-ahead parameters (i.e. k_1 and k_2). The regression models developed in Section 5.3.4. are applicable when J/M > 1.7 for k_1 and 1.7 < J/M < 7.3 for k_2. Since J/M of this example problem is 2.25 (9/4), k_1 and k_2 can be determined by applying these models to the example problem. The models need four variables, i.e. J, M, τ and R. J = 9 and M = 4 in this example problem. The value of τ and R can be evaluated from their functions:

$$\tau = \frac{1-\bar{d}}{C_{max}} \quad \text{and} \quad R = \frac{d_{max} - d_{min}}{C_{max}}$$

$$\bar{d} = \frac{15+12+7+10+10+18+20+11+11}{9} = 12.67\,; \quad d_{max} = 20; \quad d_{min} = 7$$

Makespan can be estimated by using the equation given in Section 5.3.4.

$$\begin{aligned}
\text{fsum} &= \frac{\max[1,0]+10}{1} + \frac{\max[4,0]+4+\max[4,2]+8+\max[4,2]+9+\max[4,5]+9}{4} \\
&+ \frac{\max[3,2]+5+\max[3,2]+8+\max[3,5]+8}{3} + \frac{\max[4,0]+7+\max[4,2]+9}{2} \\
&+ \frac{\max[4,0]+8+\max[4,2]+11}{2} + \frac{\max[9,2]+4+\max[9,2]+6+\max[9,5]+6}{3} \\
&+ \frac{\max[8,0]+8+\max[8,2]+10}{2} + \frac{\max[5,2]+6+\max[5,2]+9+\max[5,5]+9}{3} \\
&+ \frac{\max[5,2]+7+\max[5,2]+11+\max[5,5]+11}{3} \\
&= 11+11.75+10.67+12+13.5+14.33+17+13+14.67 \\
&= 117.92
\end{aligned}$$

$$C_{max} = \frac{\text{fsum}}{4} = 29.48$$

Therefore, τ = 1-(12.67/29.48) = 0.57 and R = (20-7)/29.48 = 0.44

The calculations to obtain k_1 and k_2 are as follows:

$$Log(k_1) = 1.8297 - 0.0326*(4) - 0.2628*(0.44) - 3.4394*(0.57) - 0.9927*(0.44)*(0.57) + 3.4555*(0.57)^2 = 0.4969$$

$$k_1 = \exp(0.4969) = 1.64$$

$$Sqrt(k_2) = 2.2707 - 0.0174*(9) - 0.0912*(4) + 0.5022*(0.57) + 0.0017*(9)*(4) - 0.0193*(9)*(0.57) = 1.9977$$

$$k_2 = (1.9977)^2 = 3.99$$

Once the appropriate values of k_1 and k_2 are determined, the ATC method can be applied to the example problem:

- Set t = 0, mt_i = [0,2,2,5], NS = [J1,J2,J3,J41,J42,J5,J6,J71,J72].
- Machine with minimum mt_i is M1. SJ = [J1,J2,J41,J42,J6]. The Priority Index is evaluated for each job listed in SJ.

Job	$\bar{p}_i$	$\bar{r}$	Priority Index
J1	$(p_{M1J2}+p_{M1J41}+p_{M1J42}+p_{M1J6})/4$ = (4+7+8+8)/4 = 6.75	$r_{M1J2}+r_{M1J41}$ $r_{M1J42}+r_{M1J6})/4$ (4+4+4+8)/4 5	$\frac{1}{10}\exp\left[-\frac{\max[15-10-0,0]}{1.64*6.75}\right]\exp\left[-\frac{\max[1-0,0]}{3.99*5}\right]$ = 0.0605
J2	$(p_{M1J1}+p_{M1J41}+p_{M1J42}+p_{M1J6})/4$ = (10+7+8+8)/4 = 8.25	$r_{M1J1}+r_{M1J41}$ $r_{M1J42}+r_{M1J6})/4$ (1+4+4+8)/4 4.25	$\frac{2}{4}\exp\left[-\frac{\max[12-4-0,0]}{1.64*8.25}\right]\exp\left[-\frac{\max[4-0,0]}{3.99*4.25}\right]$ = 0.2186
J41	$(p_{M1J1}+p_{M1J2}+p_{M1J42}+p_{M1J6})/4$ = (10+4+8+8)/4 = 7.5	$r_{M1J1}+r_{M1J2}$ $r_{M1J42}+r_{M1J6})/4$ (1+4+4+8)/4 4.25	$\frac{3}{7}\exp\left[-\frac{\max[10-7-0,0]}{1.64*7.5}\right]\exp\left[-\frac{\max[4-0,0]}{3.99*4.25}\right]$ = 0.2652
J42	$(p_{M1J1}+p_{M1J2}+p_{M1J41}+p_{M1J6})/4$ = (10+4+7+8)/4 = 7.25	$r_{M1J1}+r_{M1J2}$ $r_{M1J41}+r_{M1J6})/4$ (1+4+4+8)/4 4.25	$\frac{3}{8}\exp\left[-\frac{\max[10-8-0,0]}{1.64*7.25}\right]\exp\left[-\frac{\max[4-0,0]}{3.99*4.25}\right]$ = 0.2503
J6	$(p_{M1J1}+p_{M1J2}+p_{M1J41}+p_{M1J42})/4$ = (10+4+7+8)/4 = 7.25	$r_{M1J1}+r_{M1J2}$ $r_{M1J41}+r_{M1J42})/4$ (1+4+4+4)/4 3.25	$\frac{1}{8}\exp\left[-\frac{\max[20-8-0,0]}{1.64*7.25}\right]\exp\left[-\frac{\max[8-0,0]}{3.99*3.25}\right]$ = 0.0246

The job with the highest priority index is J41. Since J41 is a split job and $p_{M1J41} < p_{M1J42}$, J42 is selected to be assigned to M1. ST(J42,M1) = 4, CT(J42,M1) = 4 + 8 = 12, mt_{M1} = 12. At this point, J41 has to be assigned to the machine that can complete it earliest. J41 can be processed on two machines: M1 and M2. The tentative ST and CT of J41 on these machines are:

M1: ST = max [12,4] = 12, CT = 12 + 7 = 19

M2: ST = max [2,4] = 4, CT = 4 + 9 = 13

Thus, J41 is assigned to M2 with ST(J41,M2) = 4, CT(J41,M2) = 13. The difference in completion times between J41 and J42 is equal to q_{j1j2}. No revision is needed here. mt_{M2} = 13, NS = [J1,J2,J3,J5,J6,J71,J72].

- mt_i = [12,13,2,5]. The machine with minimum mt_i is M31 and t = 2. SJ = [J2,J3,J5,J71,J72] and the priority indices for these jobs are 0.1888, 0.2393, 0.1141, 0.1926, 0.1576, respectively. Since J3 obtained the highest index, it is assigned to M31 with ST(J3,M31) = [2,3] = 3 and CT(J3,M31) = 3 + 8 = 11. mt_{M31} = 11, NS = [J1,J2,J5,J6,J71,J72].
- mt_i = [12,13,11,5]. The machine with minimum mt_i is M32 and t = 5. SJ = [J2,J5,J71,J72] and the priority indices for these jobs are 0.2222, 0.1729, 0.2222, 0.1818, respectively. There is a tie between J2 and J71. As J71 is a split job, it has higher priority than J2. Since $p_{M32J72} > p_{M32J71}$, J72 is selected to be assigned to M32. ST(J72,M32) = 5, CT(J72,M32) = 16, and mt_{M32} = 16. At this point, the effort is to find a machine that can complete J71 earliest. The start times and completion times of the machines capable of processing J71 are:

  M2: ST = max [13,5] = 13, CT = 13 + 6 = 19,

  M31: ST = max [11,5] = 11, CT = 11 + 9 = 20,

  M32: ST = max [16,5] = 16, CT = 16 + 9 = 25,

  Thus, J71 is assigned to M2. Since CT(J71,M2) – CT(J72,M32) = 3 > q_{j1j2}, the start time of J72 on M32 has to be delayed such that CT(J72,M32) = 18 and ST(J72,M32) = 7. mt_{M2} = 19, mt_{M32} = 18, and NS = [J1,J2,J5,J6].
- mt_i = [12,19,11,18]. The machine with minimum mt_i is M31 and t = 11. SJ = [J2,J5] and the priority indices for these jobs are 0.2222, 0.3115, respectively. As the job with highest priority index, J5 is assigned to M31 with ST(J5,M31) = 11 and CT(J5,M31) = 17. mt_{M31} = 17 and NS = [J1,J2,J6].

- mt_i = [12,19,17,18]. The machine with minimum mt_i is M1 and t = 12. SJ = [J1,J2,J6] and the priority indices for these jobs are 0.100, 0.500, 0.125, respectively. As the job with highest priority index, J2 is assigned to M1 with ST(J2,M1) = 12 and CT(J2,M1) = 16. mt_{M1} = 16 and NS = [J1,J6].
- mt_i = [16,19,17,18]. The machine with minimum mt_i is M1 and t = 16. SJ = [J1,J6] and the priority indices for these jobs are 0.100, 0.125, respectively. As the job with highest priority index, J6 is assigned to M1 with ST(J6,M1) = 16 and CT(J6,M1) = 24. mt_{M1} = 24 and NS = [J1].
- mt_i = [24,19,17,18]. At this point, the only unscheduled job is J1, which can be processed only on M1. Thus, J1 is assigned to M1 with ST(J1,M1) = 24, CT(J1,M1) = 34 and mt_{M1} = 34.

Table 5.3 shows the summarized initial schedule and weighted tardiness obtained by applying these methods. The weighted tardiness is evaluated as a job's weight times max [due date – completion time, 0]. The total WT is the sum of the weighted tardiness of all jobs. If a solution is infeasible (i.e. some constraints are violated), the TWT would receive a penalty. As shown in Table 5.3, the solution yielded by EDD method is infeasible, i.e. the difference in completion times between J41 and J42 as two split portions of a job is larger than q_{j1j2}. Therefore, this solution receives a penalty of M, which indicates a very large number that reflects the infeasibility of a solution. The EDDsp, LFJ/LFM, and ATC methods yield feasible initial solutions with the lowest TWT obtained by the ATC method.

With the initial solution in hand, the effort to find an optimal/near-optimal solution is continued by applying the steps of tabu search documented in section 5.5. Although the ATC method yields the most superior initial solution, the initial solution generated by EDD method is selected to demonstrate the application of tabu search. This is done in order to demonstrate the capability of tabu search in using an infeasible initial solution to finally identify an optimal/near-optimal final solution. A Gantt chart of the initial solution generated by EDD method is shown in Figure 5.2.

Table 5.3 Initial solutions of example problem

Jobs	EDD			EDDsp			LFJ/LFM			ATC		
	MC	CT	WT	MC	CT	WT	MC	CT	WT	MC	CT	WT
J1	1	11	0	1	11	0	1	11	0	1	34	19
J2	31	13	2	31	13	2	1	23	22	1	16	8
J3	2	8	2	2	8	2	31	11	8	31	11	8
J41	2	24	42	2	24	42	1	18	24	2	13	9
J42	1	19	27	1	23	39	2	17	21	1	12	6
J5	31	19	2	31	19	2	32	25	14	31	17	0
J6	1	27	7	1	31	11	2	27	7	1	24	4
J71	2	15	8	2	15	8	31	20	18	2	19	16
J72	32	16	10	32	16	10	32	19	16	32	18	14
Total WT	100 + M			116			130			84		

MC = machine index, CT = job's completion time, WT = job's weighted tardiness

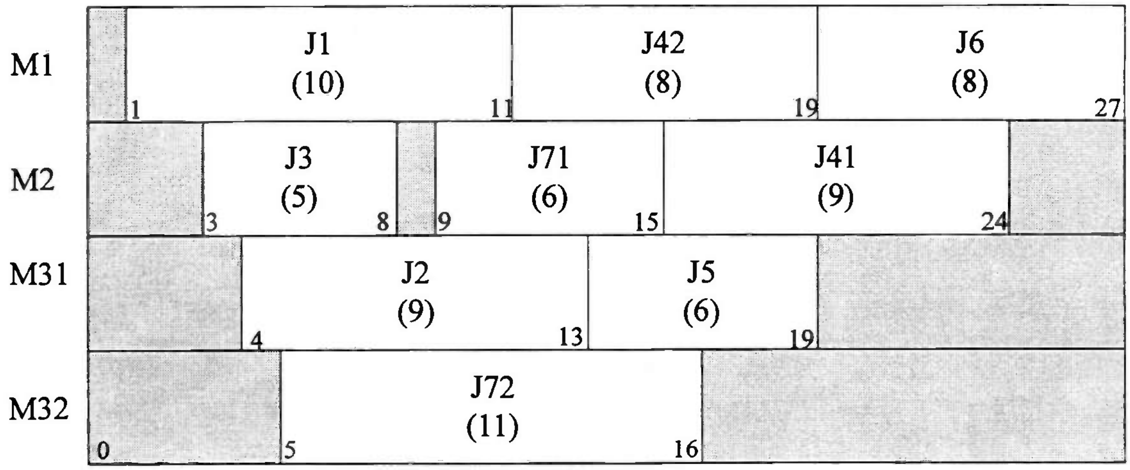


Figure 5.2 Gantt Chart for initial solution of example problem generated by applying EDD method

<u>**Step 1 & 2**</u> All possible interchange (swap) of two jobs are considered. The swap between J1 and J2 is ruled out, as J1 cannot be processed on M31. A similar situation exists for J1 with J3 and J1 with J41. The swap between J1 and J42 is feasible as both are processed on M1, J42 is not processed on M1, r_{J1} < CT(J42,M1) and r_{J42}< CT(J1,M1).

Swapping J1 and J42 results in the following changes on M1: $ST_r(J42,M1) = 4$ and $CT_r(J42,M1) = 12$, $ST_r(J1,M1) = 12$ and $CT_r(J1,M1) = 22$, $ST_r(J6,M1) = 22$ and $ST_r(J6,M1) = 30$. Note that the subscript 'r' following ST and CT denotes that they are revised. As the CT of J41 in the initial solution is 24, the JIT requirement on J41 and J42 is violated. Because the swap move is initiated by J42, a sequence change for J41 on M2 would be considered to attain feasibility. This is the first-level adjustment mentioned in section 5.4.1. J41 is moved forward so that its completion time on M2 would be 11 (i.e. 1 units less than CT(J42,M1)). However, this is only possible if J41 is released at $t = 2$. Thus, $ST_r(J41,M2) = 4$ and $CT_r(J41,M2) = 13$. The revised ST and CT for the rest of the jobs processed on M2 are: $ST_r(J3,M2) = 13$ and $CT_r(J3,M2) = 18$, $ST_r(J71,M2) = 18$ and $CT_r(J71,M2) = 24$. The JIT requirement on J71 and J72 is violated because CT(J72,M32) in the initial solution is 16. Since J71 is scheduled on the same machine as J41, a second-level adjustment is applied. J72 has to be moved forward on M32 to attain feasibility without changing the sequence. This results in $ST_r(J72,M32) = 12$ and $CT_r(J72,M32) = 23$. The resulting solution is feasible with a TWT of 108.

The swap between J1 and J5 is ruled out as J1 cannot be processed on M31. Exchanging J1 and J6 is feasible since J1 and J6 are scheduled on M1 in the initial solution, $r_{J1} < CT(J6,M1)$ and $r_{J6} < CT(J1,M1)$. Swapping J1 and J6, the new start and completion times for jobs scheduled on M1 are: $ST_r(J6,M1) = 8$ and $CT_r(J6,M1) = 16$, $ST_r(J42,M1) = 16$ and $CT_r(J42,M1) = 24$, $ST_r(J1,M1) = 24$ and $CT_r(J1,M1) = 34$. The revised completion time of J42 on M1 does not violate the JIT requirement as $CT(J41,M2) = 24$ in the initial solution. The TWT for swapping J1 with J6 is 127.

J2 and J3 is the next feasible swap move to consider. Swapping J2 and J3 results in the following changes on M2: $ST_r(J2,M2) = 4$ and $CT_r(J2,M2) = 12$, $ST_r(J71,M2) = 12$ and $CT_r(J71,M2) = 18$, $ST_r(J41,M2) = 18$ and $CT_r(J41,M2) = 27$; on M31: $ST_r(J3,M31) = 3$ and $CT_r(J3,M31) = 11$, $ST_r(J5,M31) = 11$ and $CT_r(J5,M31) = 17$. Notice that the JIT requirement on J71 and J72 is violated as CT(J72,M32) in the initial solution is equal to 16. The revised start and completion times of J72 on M32 are: $ST_r(J72,M32) = 6$ and $CT_r(J72,M32) = 17$. The JIT requirement on J41 and J42 is also violated as $CT(J42,M1) = 19$ in the initial solution. Since J41 is scheduled on the same machine as J2, a second-level adjustment is applied. The new start and completion times of the jobs processed on

M1 are: $ST_r(J1,M1) = 1$ and $CT_r(J1,M1) = 11$, $ST_r(J42,M1) = 18$ and $CT_r(J42,M1) = 26$, $ST_r(J6,M1) = 26$ and $CT_r(J6,M1) = 33$. This schedule results in a TWT of 146. 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 values.

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
J1 and J42	108	J3 and J71	175
J1 and J6	127	J41 and J42	120
J2 and J3	146	J41 and J71	116
J2 and J5	120 + M	J42 and J6	123
J2 and J71	133	J5 and J71	117
J2 and J72	108 + M	J5 and J72	114
J3 and J41	100	J71 and J72	112

Insert Moves							
Job	Machine	Position	TWT	Job	Machine	Position	TWT
J2	M1	1	130	J5	M2	2	154
J2	M1	2	118	J5	M2	3	144
J2	M1	3	146	J5	M2	4	134
J2	M1	4	158	J5	M32	1	224
J2	M2	1	214	J5	M32	2	106 + M
J2	M2	2	193	J6	M2	2	191
J2	M2	3	190	J6	M2	3	170
J2	M2	4	152	J6	M2	4	119
J2	M32	1	215	J71	M31	1	126
J2	M32	2	122 + M	J71	M31	2	124
J3	M31	1	150	J71	M31	3	130
J3	M31	2	158	J72	M31	1	148 + M
J3	M31	3	154	J72	M31	2	226
J3	M32	1	214	J72	M31	3	138
J3	M32	2	148				

Insert moves are now considered. J1 cannot be inserted to other machines as it can only be processed on M1. Inserting J2 in the first position of M1 (i.e. preceding J1) is feasible as J2 can be processed on M1 and $r_{J2} < CT(J1,M1)$. The new start and completion times of the jobs scheduled on M1 are: $ST_r(J2,M1) = 4$ and $CT_r(J2,M1) = 8$, $ST_r(J1,M1) = 8$ and $CT_r(J1,M1) = 18$, $ST_r(J42,M1) = 18$ and $CT_r(J42,M1) = 26$, $ST_r(J6,M1) = 26$ and $CT_r(J6,M1) = 34$. As CT(J41,M2) in the initial solution is 24, the JIT requirement on J41 and J42 is violated. In order to satisfy the JIT requirement, the start and completion times of J41 on M2 are revised to be $ST_r(J41,M2) = 16$ and $CT_r(J41,M2) = 25$. This is the second-level adjustment mentioned in section 5.4.2. Notice that this revision does not affect the other jobs sequenced on M2. As J2 is moved to M1, M31 is left with J5. The new start and completion times of J5 are: $ST_r(J5,M31) = 9$ and $CT_r(J5,M31) = 15$. The TWT of this schedule is 130.

The next feasible insert move is to insert J2 to the second position of M1 (i.e. between J1 and J42). The new start and completion times of the jobs scheduled on M1 are: $ST_r(J1,M1) = 1$ and $CT_r(J1,M1) = 11$, $ST_r(J2,M1) = 11$ and $CT_r(J2,M1) = 15$, $ST_r(J42,M1) = 15$ and $CT_r(J42,M1) = 23$, $ST_r(J6,M1) = 23$ and $CT_r(J6,M1) = 31$. Notice that the new completion time evaluated for J42 on M1 does not violate the JIT requirement as CT(J41,M2) is 24 in the initial solution. On M31, the new start and completion times for J5 are: $ST_r(J5,M31) = 9$ and $CT_r(J5,M31) = 15$. The TWT of this schedule is 118.

Inserting J2 to the third position of M1 (i.e. between J42 and J6) is the next feasible move. Before J2 is inserted, notice that the JIT requirement on J41 and J42 is violated in the initial solution. In order to fix this violation, the start time of J42 on M1 is delayed such that $ST_r(J42,M1) = 15$ and $CT_r(J42,M1) = 23$. The complete revised start and completion times of the jobs sequenced on M1 are: $ST_r(J1,M1) = 1$ and $CT_r(J1,M1) = 11$, $ST_r(J42,M1) = 15$ and $CT_r(J42,M1) = 23$, $ST_r(J2,M1) = 23$ and $CT_r(J2,M1) = 27$, $ST_r(J6,M1) = 27$ and $CT_r(J6,M1) = 35$. The TWT for this schedule is 146. The insert moves are continued in the same fashion for all feasible moves. The overall insert moves applied to the initial solution and their total weighted tardiness values are shown in Table 5.4.

<u>Step 3</u> The minimum TWT is 100. The move that results in this value is swapping J3 with J41. The schedule generated by swapping J3 with J41 would be used as the seed for next iteration. At this point, the following parameters need to be updated:

(1) Tabu list

The primary use of tabu list is to prevent the search from revisiting previous solutions or repeating its previous moves. Whenever a move is made, the tabu list is updated by storing the attributes of the move. In this case, J3 and J41 are the first entry in the tabu list. The presence of J3 and J41 in the tabu list implies that these jobs are not allowed to swap positions for the number iterations indicated by the size of the tabu list unless an aspiration criteria is satisfied. As mentioned in section 5.5, two types of tabu list size are used: fixed tabu list size and variable tabu list size. The tabu list size is evaluated as follows:

- For fixed tabu list size = $INT(9*\sqrt{4}/(3*\sqrt{2})) = INT(4.24) = 4$.
- For variable tabu list size:
  - The initial size = $INT(9*\sqrt{4}/(3*\sqrt{2})) = INT(4.24) = 4$
  - The decreased size = $INT(9*\sqrt{4}/(4*\sqrt{2})) = INT(3.18) = 3$
  - The increased size = $INT(9*\sqrt{4}/(2.5*\sqrt{2})) = INT(5.09) = 5$.

(2) Aspiration Level (AL)

The AL is initially set equal to the TWT of the initial solution, which is 100 + M. Since swapping J3 with J41 yields a TWT of 100, the AL is updated to be equal to 100. If a tabu move in the next iteration results in a TWT that is less than 100, 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 a local optimum. As the solution obtained by swapping J3 with J41 (i.e. S_1) is selected as the best solution, S_1 is admitted to CL. Since S_1 is better than S_0, S_1 receives a star, which indicates that it has the potential to become a local optimum. At this point, the CL has two entries and IL has one entry:

CL: { [J1/M1(1,11), J2/M31(4,13), J3/M2(3,8), J41/M2(15,24), J42/M1(11,19),
J5/M31(13,19), J6/M1(19,27), J71/M2(9,15), J72/M32(5,16)];
[J1/M1(12,22), J2/M31(4,13), J3/M2(19,24), J41/M2(4,13), J42/M1(4,12),
J5/M31(13,19), J6/M1(22,30), J71/M2(13,19), J72/M32(7,18)] }

IL: { [J1/M1(1,11), J2/M31(4,13), J3/M2(3,8), J41/M2(15,24), J42/M1(11,19),
J5/M31(13,19), J6/M1(19,27), J71/M2(9,15), J72/M32(5,16)] }

(4) Number of iterations without improvement (IT)

Initially, IT equals to zero. Since there is improvement in the TWT, i.e. a change from 100 + M to 100, the IT 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. In this case, the matrix consists of 9 x 4 cells. Initially, all 36 cells are set equal to zero. The first iteration obtained by swapping J3 with J41 results in the following entries in LTM matrix, as presented in Table 5.5.

Step 4 To terminate the search, two stopping criteria are used: ITmax and ILmax. For fixed and variable size of tabu list, the stopping criteria are evaluated as follows:

- For fixed tabu list size:

  ITmax = INT(9/6.25) = INT(1.44) = 1

  ILmax = INT(9/4) = INT(2.25) = 2

  The search is terminated if ITmax reaches 1 or ILmax reaches 2, whichever comes first.

Table 5.5 Entries into the LTM matrix after perturbing the initial solution

Job Index	M1	M2	M31	M32
J1	1	-	-	-
J2	0	0	1	0
J3	-	1	0	0
J41	0	1	-	-
J42	1	0	-	-
J5	-	0	1	0
J6	1	0	-	-
J71	-	1	0	0
J72	-	0	0	1

- For variable tabu list size:

  The ITmax and ILmax 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 $\lceil 1/3 \rceil = 1$ iteration 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 $\lceil 1/3 \rceil = 1$ iteration 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 $\lceil 1/3 \rceil = 1$ iteration 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 problem, the search is terminated after 5 iterations are made. Coincidentally, both stopping criteria are activated simultaneously, i.e. the number of iterations without improvement is equal to 1 and the number of entries into the IL has reached 2. The results of the 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	--	[J1/M1(1,11), J2/M31(4,13), J3/M2(3,8), J41/M2(15,24), J42/M1(11,19), J5/M31(13,19), J6/M1(19,27), J71/M2(9,15), J72/M32(5,16)]**	100+M	Yes
1	Swap (J3,J41)	[J1/M1(12,22), J2/M31(4,13), J3/M2(19,24), J41/M2(4,13), J42/M1(4,12), J5/M31(13,19), J6/M1(22,30), J71/M2(13,19), J72/M32(7,18)]*	100	
2	Swap (J1,J6)	[J1/M1(20,30), J2/M31(4,13), J3/M2(19,24), J41/M2(4,13), J42/M1(4,12), J5/M31(13,19), J6/M1(12,20), J71/M2(13,19), J72/M32(7,18)]*	98	
3	Insert (J3,M32, P1)	[J1/M1(20,30), J2/M31(4,13), J3/M32(5,13), J41/M2(4,13), J42/M1(4,12), J5/M31(13,19), J6/M1(12,20), J71/M2(17,23), J72/M32(13,24)]*	96	
4	Swap (J71,J72)	[J1/M1(20,30), J2/M31(4,13), J3/M32(5,13), J41/M2(4,13), J42/M1(4,12), J5/M31(13,19), J6/M1(12,20), J71/M32(13,22), J72/M2(14,21)]**	88	Yes
5	Swap (J2,J3)	[J1/M1(20,30), J2/M32(5,14), J3/M31(3,11), J41/M2(4,13), J42/M1(4,12), J5/M31(11,17), J6/M1(12,20), J71/M32(14,23), J72/M2(15,22)]	88	

The CL has 6 entries and the IL has 2 entries. The best solution obtained by employing short-term memory function is found at the fourth iteration with a TWT value of 88. The best solution is pointing to the following schedule: [J1/M1(20,30), J2/M31(4,13), J3/M32(5,13), J41/M2(4,13), J42/M1(4,12), J5/M31(13,19), J6/M1(12,20), J71/M32(13,22), J72/M2(14,21)].

<u>Step 5</u> At this point, the search can be restarted from a different region of the solution space. The restarting point is identified 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, which is 5, is J1 on M1, J41 on M2, J42 on M1, J5 on M31, and J6 on M1. Notice that J1 can only be processed on one machine (i.e. M1), thus J1 is not considered. The row-wise first best strategy points to fixing J41 on M2. Thus, the first restart solution based on maximal frequency is generated by fixing J41 on M2 in the initial solution. The first restart solution is [J1/M1(1,11), J2/M31(4,13), J3/M2(3,8), <u>J41/M2</u>(15,24), J42/M1(11,19), J5/M31(13,19), J6/M1(19,27), J71/M2(9,15), J72/M32(5,16)]. The tabu list and IT are re-initialized back to zero. The AL is reset to the TWT of the restart solution, which is equal to 100+M. 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	M1	M2	M31	M32
J1	5	-	-	-
J2	0	0	4	1
J3	-	2	1	2
J41	0	5	-	-
J42	5	0	-	-
J5	-	0	5	0
J6	5	0	-	-
J71	-	3	0	2
J72	-	2	0	3

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 4 iterations because the number of iterations without improvement has reached its maximum, which is 1 (for fixed tabu list size). Coincidentally, the number of entries into IL also reached its maximum (2). The best solution obtained from the first LTM-max restart is found at the third iteration with a TWT value of 90.

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	--	[J1/M1(1,11), J2/M31(4,13), J3/M2(3,8), J41/M2(15,24), J42/M1(11,19), J5/M31(13,19), J6/M1(19,27), J71/M2(9,15), J72/M32(5,16)]**	100+ M	Yes
1	Swap (J1,J42)	[J1/M1(12,22), J2/M31(4,13), J3/M2(13,18), J41/M2(4,13), J42/M1(4,12), J5/M31(13,19), J6/M1(22,30), J71/M2(18,24), J72/M32(12,23)]*	108	
2	Insert (J3,M32, P1)	[J1/M1(12,22), J2/M31(4,13), J3/M32(5,13), J41/M2(4,13), J42/M1(4,12), J5/M31(13,19), J6/M1(22,30), J71/M2(17,23), J72/M32(13,24)]*	98	
3	Swap (J71,J72)	[J1/M1(12,22), J2/M31(4,13), J3/M32(5,13), J41/M2(4,13), J42/M1(4,12), J5/M31(13,19), J6/M1(22,30), J71/M32(13,22), J72/M2(14,21)]**	90	Yes
4	Swap (J2,J3)	[J1/M1(12,22), J2/M32(5,14), J3/M31(3,11), J41/M2(4,13), J42/M1(4,12), J5/M31(11,17), J6/M1(22,30), J71/M32(14,23), J72/M2(15,22)]	90	

Since the total number of restart is set equal to 2, the search process is entitled to a second restart. Again, the restarting point is determined by selecting the job-machine pair with maximum frequency from the LTM matrix. The entries to LTM matrix at the termination of the first restart are shown in Table 5.9. Using the row-wise first best strategy, the maximum frequency points to fixing J42 on M1 (fixing J41 on M2 was used in the first restart). Thus, the second restart solution is generated by fixing J42 on M1 in the initial solution. The second restart solution is [J1/M1(1,11), J2/M31(4,13), J3/M2(3,8), J41/M2(15,24), J42/M1(11,19), J5/M31(13,19), J6/M1(19,27), J71/M2(9,15), J72/M32(5,16)]. The tabu list and IT 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	M1	M2	M31	M32
J1	9	-	-	-
J2	0	0	7	2
J3	-	3	2	4
J41	0	9	-	-
J42	9	0	-	-
J5	-	0	9	0
J6	9	0	-	-
J71	-	5	0	4
J72	-	4	0	5

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 4 iterations because the number of iterations without improvement has reached its maximum, which is 1 (for fixed tabu list size). Coincidentally, the number of entries into IL also reached its maximum (2). The best solution obtained from the second restart is found at the third iteration with a TWT value of 104.

Step 6 Once the entire search is terminated, the optimal/near-optimal solution is selected from the Index List as the solution with the minimum total weighted tardiness. The best solution of all is pointing to the one obtained in the initial search with a TWT of 88. The schedule that corresponds to this solution is [J1/M1(20,30), J2/M31(4,13), J3/M32(5,13), J41/M2(4,13), J42/M1(4,12), J5/M31(13,19), J6/M1(12,20), J71/M32(13,22), J72/M2(14,21)].

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	--	[J1/M1(1,11), J2/M31(4,13), J3/M2(3,8), J41/M2(15,24), J42/M1(11,19), J5/M31(13,19), J6/M1(19,27), J71/M2(9,15), J72/M32(5,16)]**	100+ M	Yes
1	Swap (J71,J72)	[J1/M1(1,11), J2/M31(4,13), J3/M2(3,8), J41/M2(15,24), J42/M1(15,23), J5/M31(13,19), J6/M1(23,31), J71/M32(5,14), J72/M2(8,15)]*	112	
2	Swap (J3,J41)	[J1/M1(12,22), J2/M31(4,13), J3/M2(20,25), J41/M2(4,13), J42/M1(4,12), J5/M31(13,19), J6/M1(22,30), J71/M32(10,19), J72/M2(13,20)]*	106	
3	Swap (J1,J6)	[J1/M1(20,30), J2/M31(4,13), J3/M2(20,25), J41/M2(4,13), J42/M1(4,12), J5/M31(13,19), J6/M1(12,20), J71/M32(10,19), J72/M2(13,20)]**	104	Yes
4	Swap (J2,J3)	[J1/M1(20,30), J2/M2(20,28), J3/M31(3,11), J41/M2(4,13), J42/M1(4,12), J5/M31(11,17), J6/M1(12,20), J71/M32(10,19), J72/M2(13,20)]	104	

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 the 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. This implies that the application of long-term memory does not improve the quality of solution obtained by the short-term memory. There are two possible reasons for this occurrence. First, the best solution obtained in the initial search is the optimal solution. Second, the approach used in the long-term memory function is not capable of directing the search to a different region. In this case, the first reason is not the right one to explain this occurrence. The best solution (88) obtained in the initial search is not the optimum, as it will be proven in Chapter 6. The second reason is the right explanation here, i.e. the maximum frequency approach is not effective enough in guiding the search to a different direction. In light of this finding, the minimum frequency approach of long-term memory function is applied.

Table 5.11 Summary of results for the entire search process based on LTM-max

Restart No.	Best solutions obtained	TWT
Initial	J1/M1(20,30), J2/M31(4,13), J3/M32(5,13), J41/M2(4,13), J42/M1(4,12), J5/M31(13,19), J6/M1(12,20), J71/M32(13,22), J72/M2(14,21)	88
First	J1/M1(12,22), J2/M31(4,13), J3/M32(5,13), J41/M2(4,13), J42/M1(4,12), J5/M31(13,19), J6/M1(22,30), J71/M32(13,22), J72/M2(14,21)	90
Second	J1/M1(20,30), J2/M31(4,13), J3/M2(20,25), J41/M2(4,13), J42/M1(4,12), J5/M31(13,19), J6/M1(12,20), J71/M32(10,19), J72/M2(13,20)	104

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 0, would be J2 on M1 if row-wise first best strategy is used. Therefore, the starting point for the first restart using the minimum frequency will be generated from the initial solution by fixing J2 on M1. In the initial solution, J2 is processed on M31. J2 has to be removed from M31 and inserted to the first position of M1, i.e. preceding J1. This insert move would cause changes in the start and completion times of the jobs processed on M1 as follows: ST_r(J2,M1) = 4 and CT_r(J2,M1) = 8, ST_r(J1,M1) = 8 and CT_r(J1,M1) = 18, ST_r(J42,M1) = 18 and CT_r(J42,M1) = 26, ST_r(J6,M1) = 26 and CT_r(J6,M1) = 34. As the JIT requirement on J41 and J42 is violated (i.e. |CT(J41,M2) – CT_r(J42,M1)| = 2 > q_{j1j2}), the start time of J41 on M2 has to be delayed such that ST_r(J41,M2) = 16 and CT_r(J41,M2) = 25. Thus, the starting point for the first restart using minimum frequency is [J1/M1(8,18), <u>J2/M1</u>(4,8), J3/M2(3,8), J41/M2(16,25), J42/M1(18,26), J5/M31(9,15), J6/M1(26,34), J71/M2(9,15), J72/M32(5,16)] with a TWT value of 130. Using this solution from LTM-min as a restarting point, the search is continued in a similar fashion as in LTM-max. Based on the LTM-min, the results obtained with the first restart are shown in Table 5.12. J2 on M1 is underlined as a sign that J2 is fixed to M1 throughout the first restart. The first restart is terminated after 5 iterations because both IT and entries into the IL have reached their maximum. The best solution from the first restart is obtained at iteration 4 with a TWT of 81.

The second restart based on LTM-min would use the information provided by the LTM matrix at the time of termination of first restart. This matrix is shown in Table

5.13. The minimum frequency, which is 0, is pointing to J2 on M2 if row-wise first best strategy is used. Thus, J2 would be fixed on M2 throughout the second restart. Since J2 is processed on M31 in the initial solution, the starting point for the second restart will be generated from the initial solution by removing J2 from M31 and inserting it to the first position of M2. The generated solution is [J1/M1(1,11), J2/M2(4,12), J3/M2(12,17), J41/M2(23,32), J42/M1(23,31), J5/M31(9,15), J6/M1(31,39), J71/M2(17,23), J72/M32(11,22)]. Using this solution, the search is restarted in the similar fashion as in LTM-max and the results are shown in Table 5.14.

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	--	[J1/M1(8,18), J2/M1(4,8), J3/M2(3,8), J41/M2(16,25), J42/M1(18,26), J5/M31(9,15), J6/M1(26,34), J71/M2(9,15), J72/M32(5,16)]**	130	Yes
1	Swap (J3,J41)	[J1/M1(16,26), J2/M1(12,16), J3/M2(19,24), J41/M2(4,13), J42/M1(4,12), J5/M31(9,15), J6/M1(26,34), J71/M2(13,19), J72/M32(7,18)]*	112	
2	Insert (J71, M31,P1)	[J1/M1(16,26), J2/M1(12,16), J3/M2(13,18), J41/M2(4,13), J42/M1(4,12), J5/M31(15,21), J6/M1(26,34), J71/M31(6,15), J72/M32(5,16)]*	94	
3	Swap (J3,J71)	[J1/M1(16,26), J2/M1(12,16), J3/M31(3,11), J41/M2(4,13), J42/M1(4,12), J5/M31(11,17), J6/M1(26,34), J71/M2(13,19), J72/M32(7,18)]*	86	
4	Insert (J6,M2, P3)	[J1/M1(16,26), J2/M1(12,16), J3/M31(3,11), J41/M2(4,13), J42/M1(4,12), J5/M31(11,17), J6/M2(19,29), J71/M2(13,19), J72/M32(7,18)]**	81	Yes
5	Swap (J5,J71)	[J1/M1(16,26), J2/M1(12,16), J3/M31(3,11), J41/M2(4,13), J42/M1(4,12), J5/M2(13,17), J6/M2(17,27), J71/M31(11,20), J72/M32(8,19)]	83	

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 better than the one obtained by the initial search. This solution has an objective function value (i.e. TWT) of 81. The Gantt Chart for this solution is shown in Figure 5.3. In Chapter 6, this solution is proven to be the optimum. Therefore, for this example problem, the minimum frequency approach is more effective in identifying an optimal/near-optimal solution than the maximum frequency approach.

Table 5.13 Entries into the LTM matrix at the end of first restart based on minimum frequency

Job Index	M1	M2	M31	M32
J1	10	-	-	-
J2	5	0	4	1
J3	-	4	4	2
J41	0	10	-	-
J42	10	0	-	-
J5	-	1	9	0
J6	8	2	-	-
J71	-	6	2	2
J72	-	2	0	8

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	--	[J1/M1(1,11), J2/M2(4,12), J3/M2(12,17), J41/M2(23,32), J42/M1(23,31), J5/M31(9,15), J6/M1(31,39), J71/M2(17,23), J72/M32(11,22)]**	214	Yes
1	Swap (J2,J41)	[J1/M1(12,22), J2/M2(24,32), J3/M2(13,18), J41/M2(4,13), J42/M1(4,12), J5/M31(9,15), J6/M1(22,30), J71/M2(18,24), J72/M32(12,23)]*	144	
2	Insert (J71, M31,P1)	[J1/M1(12,22), J2/M2(18,26), J3/M2(13,18), J41/M2(4,13), J42/M1(4,12), J5/M31(15,21), J6/M1(22,30), J71/M31(6,15), J72/M32(5,16)]*	106	
3	Swap (J3,J71)	[J1/M1(12,22), J2/M2(19,27), J3/M31(3,11), J41/M2(4,13), J42/M1(4,12), J5/M31(11,17), J6/M1(22,30), J71/M2(13,19), J72/M32(7,18)]*	100	
4	Swap (J1,J6)	[J1/M1(20,30), J2/M2(19,27), J3/M31(3,11), J41/M2(4,13), J42/M1(4,12), J5/M31(11,17), J6/M1(12,20), J71/M2(13,19), J72/M32(7,18)]**	98	Yes
5	Swap (J5,J71)	[J1/M1(20,30), J2/M2(17,25), J3/M31(3,11), J41/M2(4,13), J42/M1(4,12), J5/M2(13,17), J6/M1(12,20), J71/M31(11,20), J72/M32(8,19)]	98	

Table 5.15 Summary of results for the entire search process based on LTM-min

Restart No.	Best solutions obtained	TWT
Initial	J1/M1(20,30), J2/M31(4,13), J3/M32(5,13), J41/M2(4,13), J42/M1(4,12), J5/M31(13,19), J6/M1(12,20), J71/M32(13,22), J72/M2(14,21)	88
First	J1/M1(16,26), J2/M1(12,16), J3/M31(3,11), J41/M2(4,13), J42/M1(4,12), J5/M31(11,17), J6/M2(19,29), J71/M2(13,19), J72/M32(7,18)	81
Second	J1/M1(20,30), J2/M2(19,27), J3/M31(3,11), J41/M2(4,13), J42/M1(4,12), J5/M31(11,17), J6/M1(12,20), J71/M2(13,19), J72/M32(7,18)	98

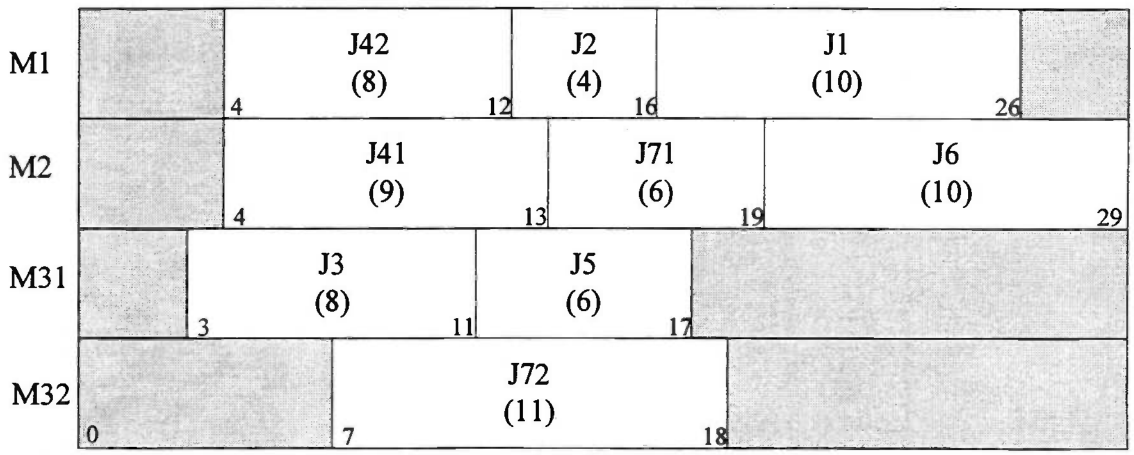


Figure 5.3 Gantt Chart for the optimal solution of the example problem

6. THE OPTIMALITY OF TABU-SEARCH BASED HEURISTIC ALGORITHM

The efficacy of the proposed heuristic algorithm is an important issue. It can be measured by the final solution and the total computation time the algorithm takes to attain it. The quality of the final solution evaluated by the heuristic can be assessed if either the optimal solution is known, or in the absence of an optimal solution, a suitable lower bound for the problem investigated is known. Referring back to the mathematical model developed in Chapter 4, an optimal solution may be obtained for small problem instances by solving the model implicitly using the branch-and-bound enumeration technique.

In order to show how a model can be formulated for a problem instance, the example problem used in Chapter 5 is used again. The model formulation follows the mathematical model developed in Chapter 4. Recall that there are two sets of binary variables, x_{ij} and $y_{ik\ell}$. The first variable, x_{ij}, receives a value of 1 if job j is assigned to machine i, or 0 otherwise. Generally, if each assignment of a job on a machine is considered, then there will be a total of n*m number of variables for x_{ij}, where n = total number of jobs and m = total number of machines. Similarly, there will be a total of n*m number of variables for c_{ij} and t_{ij}, which are two sets of real variables. In special cases where some jobs cannot be processed on some machines, one can exclude the variables that correspond to those assignments. Thus, in a real unrelated parallel machining environment, the total number of variables for x_{ij}, c_{ij}, and t_{ij} will each be typically less than n*m. The second variable, $y_{ik\ell}$, receives a value of 1 if job k precedes job ℓ on machine i, or 0 otherwise. There are a total of m*n*(n-1)/2 number of variables for $y_{ik\ell}$ if all machines are assumed to be capable of processing all jobs.

In the general model formulation for the example problem, each of the nine jobs are given the chance to be processed on each of the four machines. In this type of setting, all jobs can be processed on all machines, even though some machines are less capable than the others. This results in a total of 252 variables including 180 binary variables, and 541 constraints. This model is presented in Appendix B. In the course of formulating constraints (7) and (8) (see section 4.4 of Chapter 4), which are the JIT constraints for a pair of split jobs, the two split portions are given the chance to be processed on the same machine. This is done in order to accommodate the possibility of having a relatively large value of q_{j1j2} that would make the assignments of the split portions of a job on the same machine to be feasible. Thus, the model formulation for the example problem as presented in Appendix B is developed to provide the big picture where each machine is given the chance to process each job, and the split portions of a job are given the chance to be processed on the same machine. On the other hand, one can also formulate a more restricted (compact) model, i.e. a model that only incorporates the feasible jobs-to-machine assignments, and only allows the split portions of a job to be processed on two different machines. This type of model formulation would result in fewer variables and constraints. For the example problem, the compact model formulation would result in a total of 127 variables including 81 binary variables, and 256 constraints. The general model formulation is used in this research because it provides a comprehensive insight to a problem structure.

In order to identify the optimal solution for the example problem, its corresponding formulated model was solved using the branch-and-bound enumeration method incorporated in Hyper Lingo 4.0 (LINDO Systems, 1998) computer software. It was run on a Pentium III 450 MHz machine with 128 MB RAM. After a run time of 17 hours, 30 minutes and 46 seconds, Hyper Lingo 4.0 identified a global optimum of 81. The large amount of time that Hyper Lingo 4.0 needs to identify the optimal solution is partly due to the large number of binary variables (180) included in this model. Hyper Lingo 4.0 does not seem to be efficient enough to solve such a small problem, 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 Hyper Lingo 4.0, eleven more problem instances were generated and input to it. These problem instances were generated to be large enough for Hyper Lingo 4.0 to solve. In other words, the total number of variables and constraints of these problem instances are within the capacity of Hyper Lingo 4.0, which is 2000 constraints and 4000 variables. Except for the number of split jobs, the rest of the data for these problem instances are generated using the procedure described in section 7.1 of chapter 7. The data are presented in Table C.1 of Appendix C. Although these problem instances are sufficiently large for Hyper Lingo

4.0, their problem structures are considered small based on the categorization used in this research (see Chapter 7). The linear solver was allowed to run up to 72 hours to identify the optimal solution for each problem. The run time limit imposed here is extremely long in comparison to the run time required by the heuristic algorithm (i.e. less than 2 minutes) to solve the same problem. The results are presented in Table 6.1 Even though the run-time limit was set to 72 hours, 4 out of 11 problems (i.e. Problem Instance 8, 10, 11, 12) were not solved optimally. A feasible solution was obtained for Problem Instance 10, but it was not identified as a global optimum. The solver was not able to find any feasible solution for each of the remaining three problems when the run time reached 72 hours. The result of the experiment shows that the problem addressed in this research is highly complex. Its mathematical model is computationally difficult to solve, even when the size of the problem is small.

Table 6.1 Results of solving the problems implicitly using Hyper Lingo 4.0

Problem Instance	Problem Structure			Number of Constraints	Number of Variables		Solution	Time (sec)
	Number of Jobs	Pairs of split jobs	Number of Machines		Binary	Real		
1	8	2	4	456	144	64	380 (Opt)	7441
2	8	2	5	588	180	80	70 (Opt)	10152
3	9	2	3	396	135	54	351 (Opt)	2386
4	9	2	4	541	180	72	81 (Opt)	63046
5	10	2	3	466	165	60	450 (Opt)	34904
6	10	2	4	634	220	80	338 (Opt)	215613
7	11	2	3	542	198	66	130 (Opt)	77242
8	11	2	4	735	264	88	Infeasible	259200
9	12	2	3	624	234	72	64 (Opt)	177596
10	12	2	4	844	312	96	600 (Feas)	259200
11	15	2	6	1869	720	180	Infeasible	259200
12	17	3	5	1944	765	170	Infeasible	259200

Note: Problem Instance # 4 is the example problem used in Chapter 5. Opt implies that the solution is an optimum, while Feas implies that the solution is feasible but not optimal.

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

With the optimal solution obtained by Hyper Lingo 4.0, one can assess the quality of the solution generated by the tabu-search based heuristic algorithms. The tabu search heuristic begins with an initial solution. Referring back to Chapter 5, four different methods to generate the initial solution were developed. These methods will be referred to as IS1 for EDD method, IS2 for EDDsp method, IS3 for LFJ/LFM method, and IS4 for ATC method. The initial solution generated by each of these methods is used as a starting point for the tabu-search based heuristic. Tabu search has a 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 24 heuristic combinations. Each combination is tested on 8 problem instances presented in Table 6.1. The remaining four problem instances are not used since Hyper Lingo 4.0 failed to identify the optimal solutions 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

Hyper Lingo 4.0. 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 generate 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 Instance	TS1				TS2				TS3			
	IS1	IS2	IS3	IS4	IS1	IS2	IS3	IS4	IS1	IS2	IS3	IS4
8J*2SP*4M	2.1	2.1	2.1	2.6	2.1	0	2.1	2.1	0	2.1	2.1	2.6
8J*2SP*5M	2.9	0	0	18.6	0	0	0	17.1	2.9	0	0	18.6
9J*2SP*3M	0.6	1.7	3.7	1.7	0.6	1.7	0	0.6	0.6	1.7	3.7	1.7
9J*2SP*4M	8.6	8.6	14.8	0	8.6	8.6	1.2	0	0	0	14.8	0
10J*2SP*3M	7.8	7.8	11.1	7.8	7.8	0	0	7.8	6.7	6.7	7.8	0
10J*2SP*4M	0	0	0	0	0	0	0	0	0	0	0	0
11J*2SP*3M	6.2	0	6.2	40	0	0	6.2	6.2	6.2	0	6.2	6.2
12J*2SP*3M	3.1	3.1	23.4	78.1	3.1	3.1	3.1	3.1	3.1	3.1	23.4	70.3
Average	3.91	2.91	7.66	18.60	2.78	1.68	1.57	4.61	2.43	1.70	7.24	12.42
Problem Instance	TS4				TS5				TS6			
	IS1	IS2	IS3	IS4	IS1	IS2	IS3	IS4	IS1	IS2	IS3	IS4
8J*2SP*4M	2.1	2.1	2.1	2.6	2.1	0	2.1	2.1	0	2.1	2.1	2.6
8J*2SP*5M	2.9	0	0	18.6	0	0	0	17.1	2.9	0	0	18.6
9J*2SP*3M	0.6	1.7	3.7	1.7	0.6	1.7	0	0.6	0.6	1.7	3.7	1.7
9J*2SP*4M	8.6	8.6	14.8	0	8.6	8.6	1.2	0	0	0	14.8	0
10J*2SP*3M	7.8	7.8	11.1	7.8	7.8	0	0	0	6.7	6.7	0	0
10J*2SP*4M	0	0	0	0	0	0	0	0	0	0	0	0
11J*2SP*3M	6.2	0	6.2	6.2	0	0	6.2	6.2	6.2	0	6.2	6.2
12J*2SP*3M	3.1	3.1	23.4	78.1	3.1	3.1	0	3.1	3.1	3.1	23.4	70.3
Average	3.91	2.91	7.66	14.37	2.78	1.68	1.18	3.63	2.43	1.70	6.27	12.42

Note: J = jobs, SP = pairs of split jobs, M = machines

The average percentage deviation of all 24 heuristic combinations is 5.4% with six of the heuristic combinations below 2%. From the 24 heuristic combinations, IS3/TS5 appears to be the most effective heuristic combination in identifying the optimal solutions. The average percentage deviation for IS3/TS5 is 1.18%. The next best performer is IS3/TS2, which has an average percentage deviation of 1.57%. Both of these heuristic combinations use LFJ/LFM method as the initial solution generation method, and long-term memory with maximum-frequency strategy in the tabu-search based heuristic. Both of these heuristic combinations only take approximately 12 seconds to complete compared to 20 hours, which is the average time that Hyper Lingo 4.0 takes to find the optimal solutions for all eight problem instances. Based on the average percentage deviation, one may conjecture that the combination of IS3 and TS5 is the most effective heuristic combination in identifying optimal or near optimal solutions for the small problem structure.

Table 6.4 Computation time of the heuristics for small problems (in seconds)

Problem Instance	TS1				TS2				TS3			
	IS1	IS2	IS3	IS4	IS1	IS2	IS3	IS4	IS1	IS2	IS3	IS4
8J*2SP*4M	4.3	4.7	6.3	5.6	7.3	8.7	12.6	9.7	7.4	8.3	10.8	8.5
8J*2SP*5M	5.9	4.6	4.5	5.3	10.8	9.7	9.1	11.6	8.7	9.3	8.6	9.9
9J*2SP*3M	6.6	5.3	6.4	5.1	10.3	9.2	10.5	8.2	9.5	9.4	12.9	8.2
9J*2SP*4M	4.7	4.7	4.1	3.6	8.5	9.0	8.1	5.8	8.4	8.0	7.8	6.9
10J*2SP*3M	6.5	6.6	5.6	5.8	12.8	14.2	11.5	10.5	12.9	13.6	13.1	11.5
10J*2SP*4M	5.8	6.4	8.2	8.1	11.4	12.0	13.1	12.3	13.8	12.6	15.5	10.2
11J*2SP*3M	9.0	8.0	6.3	6.2	18.4	16.4	13.6	12.4	16.7	16.5	12.7	12.5
12J*2SP*3M	11.2	9.5	8.1	5.7	24.6	21.6	18.3	12.0	19.7	19.3	17.6	11.7
Average	6.74	6.21	6.18	5.65	13.00	12.60	12.09	10.32	12.13	12.13	12.37	9.92
Problem Instance	TS4				TS5				TS6			
	IS1	IS2	IS3	IS4	IS1	IS2	IS3	IS4	IS1	IS2	IS3	IS4
8J*2SP*4M	5.2	3.8	4.4	3.6	8.2	7.0	8.8	7.0	7.3	8.7	10.3	9.4
8J*2SP*5M	4.9	5.5	5.3	5.4	9.7	11.5	9.6	9.7	10.2	9.2	9.0	9.3
9J*2SP*3M	4.5	4.7	5.2	4.8	8.0	8.5	8.6	7.7	8.2	7.8	7.5	6.5
9J*2SP*4M	5.3	5.2	4.0	3.6	9.2	9.2	8.0	6.5	8.0	7.9	7.4	6.3
10J*2SP*3M	6.6	6.3	6.2	6.3	13.8	13.4	13.1	11.9	12.5	11.4	13.2	10.9
10J*2SP*4M	6.3	6.2	5.8	5.2	11.8	11.5	10.8	9.0	12.0	12.1	12.5	10.7
11J*2SP*3M	9.8	9.5	7.8	8.5	20.3	20.5	16.8	15.7	18.6	19.0	13.9	14.8
12J*2SP*3M	9.5	9.6	8.0	6.4	24.2	22.9	18.6	13.8	21.0	21.5	17.3	13.8
Average	6.51	6.34	5.83	5.47	13.15	13.06	11.78	10.16	12.22	12.19	11.38	10.22

Note: J = jobs, SP = pairs of split jobs, M = machines

The average computation time of all 24 heuristic combinations is 9.9 seconds. This is tremendously short in comparison to the computation time Hyper Lingo 4.0 takes to find the optimal solutions, which is 20 hours on average over 8 problem instances. Within a level of TS, IS4 appears to be fastest among the four levels of IS. The superiority of IS4 from the other levels of IS is later confirmed in a statistical analysis explained in chapter 7. Comparing the levels of TS, the computation times for TS1 and TS4 turn out to be shorter than the computation times for TS2, TS3, TS5 and TS6. This is due to the fact that TS1 and TS4 use the short-term memory of tabu search, while TS2, TS3, TS5, TS6 use the long-term memory.

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

The branch-and-bound enumeration technique is not efficient enough to solve for the optimal solution when the size of a problem structure grows larger. The branch-and-bound enumeration technique is not capable of identifying the optimal solutions for four problem instances mentioned in the previous section although these problems are categorized as small. For medium and large problem instances, the effectiveness of the heuristics can be assessed if a suitable lower bound for the problem investigated is known. However, the problem structure does not seem to lend itself to conveniently identify a lower bound. An alternative way to assess the effectiveness of the heuristics for medium and large problems 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 be evaluated by measuring how much their TWT deviates from the optimum which is zero. A problem instance with an optimal TWT of zero is generated by 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. In the process of the random assignment, care should be taken to ensure that split jobs satisfy the JIT requirement. Record the completion times of all jobs.

3. Set the due dates of all non-split jobs equal to their completion time. The due dates for the split portions of a job should be set equal to the largest completion time of the two split portions.

For the medium problem structure, 5 problem instances are generated using the above procedure. The choice of only 5 problem instances is based on two reasonings. First, there are 24 heuristic combinations (4 levels of IS and 6 levels of TS) to be tested. With 5 problem instances, the actual number of problem tested is 120 (24*5). Second, each heuristic combination requires 3 minutes to 3 hours of computation time to solve a medium problem instance. Thus, solving 5 problem instances of medium size with all heuristic combinations would need fairly large computational effort.

The combinations of IS1 – IS4 and TS1 – TS6 are applied to each problem instance. Once the values of the TWT from the heuristic combinations are obtained, they are compared to the TWT of the optimal schedule, which is zero. However, an evaluation for percentage deviation is not possible since that would lead to a division by zero. To overcome this problem, the point of reference, which is the TWT of the optimal schedule, must be shifted to a positive value. This can be accomplished by delaying the completion times of all jobs in the optimal schedule by one unit of time and thus, the TWT would be greater than zero. Generally, this results in a TWT that is equal to the sum of the weights of all jobs. This generalization only holds true if the split portions of a job are completed at the same time in the optimal schedule. If the completion times of the split portions of a job are not the same in the optimal schedule, one of the two split portions will not be tardy and thus the weight of this split portion should not be included in the evaluation for the TWT. This TWT is used as a new reference point in evaluating the percentage deviation of the solutions obtained by the heuristics. 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 five 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. Almost half of the heuristic combinations are able to identify the true optimal TWT of zero for the problem instance with 25 jobs and 10 machines. Most of the heuristic combinations obtained TWT that are less than the reference point, except in the problem instance with 45 jobs and 6 machines.

Table 6.5 Results of applying the heuristics to medium problem structures with zero values of TWT

25 Jobs, 10 Machines (Reference Point = 53)													
TWT	TS1	TS2	TS3	TS4	TS5	TS6	% Dev	TS1	TS2	TS3	TS4	TS5	TS6
IS1	4	4	4	4	4	0	IS1	0.0	0.0	0.0	0.0	0.0	0.0
IS2	4	4	4	4	4	0	IS2	0.0	0.0	0.0	0.0	0.0	0.0
IS3	4	3	0	0	0	0	IS3	0.0	0.0	0.0	0.0	0.0	0.0
IS4	4	0	0	0	0	0	IS4	0.0	0.0	0.0	0.0	0.0	0.0
30 Jobs, 9 Machines (Reference Point = 77)													
TWT	TS1	TS2	TS3	TS4	TS5	TS6	% Dev	TS1	TS2	TS3	TS4	TS5	TS6
IS1	10	10	10	10	10	10	IS1	0.0	0.0	0.0	0.0	0.0	0.0
IS2	14	14	14	14	14	14	IS2	0.0	0.0	0.0	0.0	0.0	0.0
IS3	49	11	11	11	11	11	IS3	0.0	0.0	0.0	0.0	0.0	0.0
IS4	24	24	24	24	24	24	IS4	0.0	0.0	0.0	0.0	0.0	0.0
35 Jobs, 8 Machines (Reference Point = 78)													
TWT	TS1	TS2	TS3	TS4	TS5	TS6	% Dev	TS1	TS2	TS3	TS4	TS5	TS6
IS1	69	36	11	69	36	11	IS1	0.0	0.0	0.0	0.0	0.0	0.0
IS2	1	1	1	1	1	1	IS2	0.0	0.0	0.0	0.0	0.0	0.0
IS3	11	11	11	11	11	11	IS3	0.0	0.0	0.0	0.0	0.0	0.0
IS4	97	63	93	97	63	93	IS4	24.4	0.0	19.2	24.4	0.0	19.2
40 Jobs, 7 Machines (Reference Point = 86)													
TWT	TS1	TS2	TS3	TS4	TS5	TS6	% Dev	TS1	TS2	TS3	TS4	TS5	TS6
IS1	57	57	57	57	57	57	IS1	0.0	0.0	0.0	0.0	0.0	0.0
IS2	45	45	45	45	45	45	IS2	0.0	0.0	0.0	0.0	0.0	0.0
IS3	47	47	47	47	47	47	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
45 Jobs, 6 Machines (Reference Point = 106)													
TWT	TS1	TS2	TS3	TS4	TS5	TS6	% Dev	TS1	TS2	TS3	TS4	TS5	TS6
IS1	153	149	153	153	149	153	IS1	44.3	40.6	44.3	44.3	40.6	44.3
IS2	153	149	153	153	149	153	IS2	44.3	40.6	44.3	44.3	40.6	44.3
IS3	225	225	174	225	225	174	IS3	112.3	112.3	64.2	112.3	112.3	64.2
IS4	191	184	138	191	184	138	IS4	80.2	73.6	30.2	80.2	73.6	30.2

To view the performance of each heuristic combination, the average percentage deviation over the five problem instances is evaluated and presented in Table 6.6. Four heuristic combinations, i.e. IS1/TS2, IS2/TS2, IS1/TS5, and IS2/TS5, appear to have the same minimum average percentage deviation of 8.11%. The percentage deviation averaged over the five problem instances and the 24 heuristic combinations is 12.9%. Based on these results, one may conjecture 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 Solution	Tabu-Search Based Heuristics					
Generation Method	TS1	TS2	TS3	TS4	TS5	TS6
IS1	8.87	8.11	8.87	8.87	8.11	8.87
IS2	8.87	8.11	8.87	8.87	8.11	8.87
IS3	22.45	22.45	12.83	22.45	22.45	12.83
IS4	20.91	14.72	9.88	20.91	14.72	9.88

A similar effort is made to assess the effectiveness of the heuristics in identifying optimal solutions for large problem structure. This time, four problem instances that range from 50 to 60 jobs and 11 to 15 machines are generated. The problem structure that falls within this range is considered large problem in this research. Considering the computation time required by each heuristic to solve a large problem is between 4 – 14 hours, and there are a total of 24 heuristic combinations, testing the heuristics on many problem instances can take up a large computational effort. Therefore, only four problem instances are used. The random assignment procedure that was used for medium problem structure is now applied to the large problem structure. For each problem instance, an optimal schedule with zero value of TWT is obtained by setting the due dates equal to the completion times of the jobs. All 24 heuristic combinations are applied to each problem instance and the TWT of the final solutions is evaluated accordingly. The percentage deviation of the final solutions is also 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. In the problem instance with 50 jobs and 11 machines, four of the heuristic combinations are able to identify solutions with TWT that is very close to zero. All of the heuristic combinations are able to identify solutions that are smaller than the reference point in two problem instances.

Table 6.7 Results of applying the heuristics to large problem structures with zero values of TWT

50 Jobs, 11 Machines (Reference Point = 111)												
TWT	TS1	TS2	TS3	TS4	TS5	TS6	% Dev	TS1	TS2	TS3	TS4	TS5	TS6
IS1	31	27	31	31	31	31	IS1	0.0	0.0	0.0	0.0	0.0	0.0
IS2	31	27	31	31	31	31	IS2	0.0	0.0	0.0	0.0	0.0	0.0
IS3	23	11	13	23	11	13	IS3	0.0	0.0	0.0	0.0	0.0	0.0
IS4	34	4	2	34	4	2	IS4	0.0	0.0	0.0	0.0	0.0	0.0
53 Jobs, 13 Machines (Reference Point = 126)												
TWT	TS1	TS2	TS3	TS4	TS5	TS6	% Dev	TS1	TS2	TS3	TS4	TS5	TS6
IS1	217	85	80	217	85	80	IS1	72.2	0.0	0.0	72.2	0.0	0.0
IS2	93	85	93	93	85	93	IS2	0.0	0.0	0.0	0.0	0.0	0.0
IS3	148	148	97	148	148	97	IS3	17.5	17.5	0.0	17.5	17.5	0.0
IS4	115	93	110	115	93	110	IS4	0.0	0.0	0.0	0.0	0.0	0.0
58 Jobs, 12 Machines (Reference Point = 136)												
TWT	TS1	TS2	TS3	TS4	TS5	TS6	% Dev	TS1	TS2	TS3	TS4	TS5	TS6
IS1	165	137	165	173	137	173	IS1	21.3	0.7	21.3	27.2	0.7	27.2
IS2	131	109	131	131	109	131	IS2	0.0	0.0	0.0	0.0	0.0	0.0
IS3	226	187	189	226	187	189	IS3	66.2	37.5	39.0	66.2	37.5	39.0
IS4	159	132	159	159	132	159	IS4	16.9	0.0	16.9	16.9	0.0	16.9
60 Jobs, 15 Machines (Reference Point = 144)												
TWT	TS1	TS2	TS3	TS4	TS5	TS6	% Dev	TS1	TS2	TS3	TS4	TS5	TS6
IS1	54	18	23	54	18	23	IS1	0.0	0.0	0.0	0.0	0.0	0.0
IS2	30	25	29	30	25	29	IS2	0.0	0.0	0.0	0.0	0.0	0.0
IS3	68	68	55	68	68	55	IS3	0.0	0.0	0.0	0.0	0.0	0.0
IS4	55	25	38	55	25	38	IS4	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.

As seen from this table, the average percentage deviation evaluated for all tabu-search based heuristic that use IS2 as the initial solution generation method is zero. This means that the combination of IS2 and any TS is capable of identifying solutions that are smaller than the reference point. IS4/TS2 and IS4/TS5 also obtain an average percentage deviation of zero. The percentage deviation averaged over the four problem instances and the 24 heuristic combinations is 6.9%. Based on these results, it can be concluded that the heuristics have been very effective in identifying very good near optimal solutions for the large problem structure.

Table 6.8 Average percentage deviation of the solutions obtained by the heuristics for large problem structure

Initial Solution	Tabu-Search Based Heuristics					
Generation Method	TS1	TS2	TS3	TS4	TS5	TS6
IS1	23.39	0.18	5.33	24.86	0.18	6.80
IS2	0.00	0.00	0.00	0.00	0.00	0.00
IS3	20.91	13.74	9.74	20.91	13.74	9.74
IS4	4.23	0.00	4.23	4.23	0.00	4.23

7. RESULTS AND DISCUSSIONS

Recall from Chapter 6, the tabu-search based heuristic algorithms are proven to be highly efficient in comparison to the implicit enumeration technique (namely branch-and-bound) in solving small problem structures. Each heuristic algorithm only takes an average of 10 seconds to solve the problem, while the branch-and-bound technique embedded in Hyper Lingo 4 takes an average of 20 hours. Furthermore, the quality of the solutions generated by these heuristic algorithms deviates 5.4% in average from the optimum. One of the algorithms, i.e. IS3/TS5 obtained solutions that deviate only 1.18% in average from the optimal solutions. For medium and large problem structures, the effectiveness of the algorithms was assessed by solving problem instances that are constructed to have zero total weighted tardiness. The average percentage deviation evaluated over all heuristic algorithms is 12.9% for medium problem structure and 6.9% for large problem structure. Based on these results, the tabu-search based heuristic algorithms can be conjectured to provide very good near optimal solutions, 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 the initial solution generation methods. Precisely, the intent of the 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, and total number of machines, m. Based on the size of problems used by Yaghubian et al. (1999), the size of the problem structures covered in this research is defined as follows:

Small size: up to 20 jobs and 5 machines,

Medium size: 21 – 45 jobs and 6 – 10 machines,

Large size: 46 – 60 jobs and 11 – 15 machines.

These sizes are selected to cover a wide variety of scheduling problems encountered in industry practice, and whether the computation time required to solve them using the algorithms lies within reasonable expectations. Most of the small problem structures can be solved in less than 1.5 minutes. The medium problem structures require 3 minutes to

3 hours of computation time. Solving a problem structure as large as 60 jobs and 15 machines may take as long as 14 hours. The increase in the computational time is due to the increase in the complexity of the problem, presented in the form of an enlarged search space. The increase in search space has caused the algorithm to consider more neighborhood solutions before selecting the best solution and then applying the move that results in that best solution. The increase in search space also delays the termination of the search as more moves are required before the stopping criteria is activated.

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 four 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.
4. To analyze the impact of the features incorporated in the tabu search, in particular the tabu list size and the memory function, on each size of the problem structure.

To address these research issues, a multi-factor experiment based on split-plot design is considered. The design of this experiment is explained in detail in section 7.2.

7.1. Data Generation

As mentioned before, the total number of jobs and machines involved defines the structure of a problem. In the experiment, the data for each job and machine, namely the job processing time, job weight, job release time, job due date, and machine availability are generated using randomization procedure. The notation used for the total number of jobs is n and the total number of machines is m. The procedure used to generate the data for each problem instance is documented as follows:

(1) The total number of split jobs, sn, in a problem instance is determined to be equal to $0.25n$. Since split jobs have to be in pairs, the number of split jobs should be even.

If $0.25n$ results in a decimal number, round the value to the nearest even number. If $0.25n$ results in an odd number, round up the value to the nearest even number.

(2) To determine the jobs that will receive split status, i.e. split jobs, a set of random numbers that are uniformly distributed over the interval [0,1] is generated. The total number of random numbers in the set is equal to $n - ½*sn$ (sn is evaluated in step (1)). Count the quantity of random numbers in the set that has value ≤ 0.25. If the total count is not equal to $½*sn$, generate a new set of random numbers and repeat the count. If the total count is equal to $½*sn$, assign an index number of $1,2,3,\ldots\ n - ½*sn$ to each random number in the set. As the index numbers are later used as indices for the jobs, the index numbers that are assigned to random numbers with value $\leq$ 0.25 become the indices of the first split portions of the jobs with '1' added as the last digit of the indices. The second split portions of the jobs are assigned the same indices as the first split portions but the last digit of the indices is '2'. For example, if the random number with value $\leq$ 0.25 is assigned an index of 6, then the first split portion of the job has an index of J61 and the second split portion J62.

(3) The maximum permissible limit for the difference in completion time between two split portions of a job, q_{j1j2}, is equal to 1 time unit.

(4) Three levels (types) of machine capability are included in a problem instance: the least, medium and most capable machines. The subsequent statements explain how the type of the machines and the total unit for each machine type is determined. Initially, three machine indices (M1, M2 and M3) are developed. Three random numbers are generated from a uniform distribution over the interval [1,10]. These random numbers are used as coefficients of machine capability, α_i, for each machine type i. The first α_i is assigned to M1, the second to M2 and so on. The machine that receives the smallest α_i is referred to as the most capable machine. Consequently, the machine that receives the largest α_i is referred to as the least capable machine. If m is larger than 3, generate $m - 3$ random numbers that are uniformly distributed over the interval [0,1]. Count the quantity of random numbers that have value less than or equal to 1/3, between 1/3 and 2/3, and larger than 2/3. The smallest of the three counts is the additional units for the most

capable machine. This means that if the smallest of the three counts equals to zero, the most capable machine does not have any additional unit. The largest of the three counts is the additional units for the least capable machine. Consequently, the machine with medium capability will have additional units equal to the second largest of the three counts.

(5) The least, medium and most capable machines are assumed to have the potential to process 50%, 70% and 85% of all jobs, respectively. These job percentage is noted as β_i for each machine type i. β_i is used to determine whether job j can be processed on a machine type i or not. First, a uniformly distributed random number RN in [0,1] is generated. If $RN > \beta_i$, then job j is assigned infinite processing time on machine type i. If $RN \leq \beta_i$, the processing time of job j on machine type i is determined in step (6). If the split portion of a job was assigned infinite processing time on machine type i, then its other split portion would also receive an infinite processing time on the same type of machine.

(6) The processing times are uniformly distributed over the interval $[\alpha_i + 1, \alpha_i + 20]$ for non-split jobs and $[\alpha_i + 11, \alpha_i + 20]$ for split jobs. The processing times of a job are the same on machine units of the same type.

(7) The release times of the jobs are generated from Poisson distributed random numbers with mean interarrival rate of 5. Poisson distribution was used to generate job release time by Schutten and Leussink (1996). These random numbers must take integer values.

(8) Machine availability time is generated from Poisson distributed random numbers with interarrival rate of 5. Suresh and Chudhuri (1996) used Poisson distribution to model the occurrence of machine non-availability. These random numbers must take integer values.

(9) Job weight is generated from uniformly distributed random numbers over the interval [1,4]. These random numbers must be integers.

(10) The due dates of the jobs are generated from a composite uniform distribution based on the user-defined due date range factor (R) and the due date tightness factor (τ). A random number RN from a uniform distribution over the interval [0,1] is generated. If $0 \leq RN \leq \tau$, the due date is generated from uniformly distributed

random numbers over the interval $[\bar{d} - R\bar{d}, \bar{d}]$. If $RN > \tau$, the due date is generated from uniformly distributed random numbers over the interval $[\bar{d}, \bar{d} + (C_{max} - \bar{d})R]$. The due dates must be integer values. The evaluation for C_{max} and $\bar{d}$ is described in Section 5.3.4.

Notice that all random numbers in the procedure above are generated from uniform distribution except for job release time and machine availability time, which are generated from Poisson distribution. The reasoning for the different types of distribution used is that uniform distribution is appropriate to model a length or duration of a process, while Poisson distribution is the appropriate distribution to model the occurrence of an event at a point of time.

7.2. Design of Experiment

To address research questions 1,2, and 3, a multi-factor experimental design is employed. Two performance measures are used: the total weighted tardiness and the total computation time of the algorithms. Two factors are used in the experiment, they are the initial solution generation methods (IS) and the different types of tabu-search based heuristics (TS). As described in Chapter 6, there are four different levels of IS and six different levels of TS.

In the beginning of this chapter, three different sizes of problem structures were defined and discussed. Within each size, there are different structures to consider. Within a problem structure, one can generate different problem instances (test problems) using the procedure documented in section 7.1. Since one problem instance is different from another, an experiment that involves various problem instances and various problem structures will collect large variability of results. The 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 problem instances. Thus, the differences in the performance of the algorithms, if identified, can be wholly attributed to the effect of the algorithms and not to the difference between problem instances.

All 24 (4 levels of IS * 6 levels of TS) combinations of both factors are tested in each block. At this point, the experimental design looks like a randomized complete block design. However, it is not possible to completely randomize the order of the factor combinations applied to a block as required in a randomized complete block design. Therefore, a split-plot design is selected in which IS is the whole plot treatment and TS is the subplot treatment. TS is considered as subplot treatment because it is the factor posing the maximum interest in the design. For further details on randomized complete block design and split plot design, refer to the text by Montgomery (1991).

The experiment includes all three sizes of problem structures. For small size category, three different problem structures are used; they are 9 jobs and 4 machines, 12 jobs and 3 machines, and 17 jobs and 5 machines. Another three problem structures are used under medium size category: 25 jobs and 10 machines, 35 jobs and 8 machines, and 45 jobs and 6 machines. For large size category, the types of problem structures are reduced to two. This reduction is due to its extensive computation time as explained in the beginning of this chapter. The two problem structures used for large size category are: 50 jobs and 11 machines, and 60 jobs and 15 machines.

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 characteristic of the due date, 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 Pentium III 450 MHz machines with 128 MB 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. In these tables, the total weighted tardiness is the final best solution obtained by a tabu-search based 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 be focused on the total weighted tardiness first and then on the computation time.

Table 7.1 Summary of experimental results

Performance Measure		Average Total Weighted Tardiness			Average Computation Time (in seconds)		
Problem Structure		Small	Medium	Large	Small	Medium	Large
Levels of IS	IS1	258.76	772.02	953.3	41.7	1810	15803
	IS2	259.42	774.33	937.4	37.85	1741	16268
	IS3	261.47	778.84	936.8	46.23	2389	18004
	IS4	261.93	768.77	937.2	30.35	1355	11242
Levels of TS	TS1	264.34	782.93	955.8	13.6	781	7716
	TS2	258.38	769.70	933.9	42.3	2228	20979
	TS3	259.24	768.48	932.8	43.5	2226	19617
	TS4	263.97	782.42	954.0	18.7	804	6466
	TS5	257.22	769.93	937.2	60.0	2519	18738
	TS6	259.24	767.48	933.4	56.1	2384	18460

Note: The average total weighted tardiness for each level of IS/TS is obtained by taking the average of total weighted tardiness over all blocks and all levels of TS/IS. The average computation time is evaluated in the same way as average total weighted tardiness.

7.3.1. Total Weighted Tardiness

As the summary of results only shows the average of the total weighted tardiness (TWT), one cannot conclusively say that the level of factors that has the minimum average TWT is, in a statistical sense, better than the rest. In order to perform a statistical analysis on the TWT, a preliminary data exploration is necessary to examine the distribution of TWT. It is widely known that the statistical analysis methods such as t-test are a powerful tool if the data is normally distributed. Graphic tools such as box plot is very useful to detect any departure from the assumption of normal distribution. The box plots of the TWT for all levels of IS and all levels of TS are shown in Figure F.1 - F.3 of Appendix F for small, medium and large problem structures. These box plots are generated by STATGRAPHICS Plus version 3.0 (Statistical Graphics, 1994-1997) statistics software. The plots show that the data distribution is highly skewed and long-tailed, which implies a severe departure from the normality assumption. This is due to the big discrepancy between the values of the TWT as a result of using different combinations of due date tightness factor (τ) and due date range factor (R). A problem instance that has a small τ and large R, or small τ and small R would tend to yield a relatively small or even zero value of TWT. On the other hand, a problem instance with large τ and small R would tend to yield a relatively large value of TWT. Furthermore, the TWT cannot be normally distributed since the values of TWT are integers (discrete) while normal distribution is a continuous distribution. Due to the non-normality of the data distribution, parametric methods such as F-test and t-test are not appropriate for analyzing the experimental results.

The alternative to F-test and t-test are non-parametric methods known as Friedman test and Wilcoxon signed-rank test. Friedman test is useful to check if there is any significant difference between the treatment (factor) levels. If there is an evidence of significant difference between the treatment levels, Wilcoxon signed-rank test will be applied to identify which treatment level performs distinguishably better than the rest. Friedman test utilizes rank transformation that is applied to the response variable (i.e. TWT) and a test statistic is evaluated on the ranks. Originally, Friedman test was used for analysis in single factor, randomized block experiment. After small modifications of the procedure, the test can be used to test for main effects and interactions in multi-factor experiments involving randomized block design. For a detailed description on the application of Friedman test, refer to the text by Conover (1999). The description on the modifications to the procedure can be found in the texts by Bradley (1968), and Neave and Worthington (1988).

For each size of problem structure, Friedman tests are applied to test three different hypotheses as stated below:

Hypothesis 1. H_0: There is no difference in the TWT obtained for the problem instances using the four initial solution generation methods (IS).

H_1: At least one of the initial solution generation methods tends to yield 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 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 Friedman tests are summarized in Table 7.2. With $\alpha = 0.05$, there is no significant difference between the levels of IS for all sizes of problem structure. On the contrary, there is strong significant difference between the levels of TS for all sizes of problem structure.

Table 7.2 Summary of results from Friedman tests

Problem Structure	Hypotheses 1 (IS)		Hypotheses 2 (TS)		Hypotheses 3 (IS*TS)	
	Test Statistics	p-value	Test Statistics	p-value	Test Statistics	p-value
Small	0.638	0.8876	31.574	< 0.0001	25.938	0.0387
Medium	0.891	0.8277	42.094	< 0.0001	17.846	0.2708
Large	2.576	0.4616	26.309	< 0.0001	24.853	0.0520

It is pertinent to check if the interaction effect between the levels of IS and TS is significant. If the interaction between IS and TS is significant, the effect of TS may be obscured by the interaction effect. For a significance level of 0.05, the interaction between IS and TS is significant for small problem structure. Thus, Wilcoxon signed-rank tests are applied to identify which combinations of IS and TS that differ significantly. For a detailed description on the application of the Wilcoxon signed-rank test, refer to Conover (1999). For small problem structure, there are a total 276 comparisons between all possible pairs of 24 combinations of IS and TS. Only 2 out of the 276 comparisons turn out to be significantly different, they are: IS4/TS1 with IS2/TS5, and IS4/TS4 with IS2/TS5. Since the other combinations of IS and TS do not show any significant difference, then it is safe to make pairwise comparisons just between the levels of main effects (i.e. IS or TS). Since there is no significant difference between the levels of IS (the p-value of the tests are > 0.05) for all sizes of problem structure, then the pairwise comparisons are conducted only between the levels of TS. The comparisons between the levels of TS are necessary in order to identify which level of TS, without the effect of IS, performs significantly better. This is done by applying Wilcoxon signed-rank tests on the TWT between different levels of TS. The results are shown in Table F.1 of Appendix F.

7.3.2. Computation Time

Recall from the previous section that the Friedman tests show that the TWT between the levels of IS is not significantly different. Thus, the performance of each level of IS is now 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 the computation time are shown in Figures G.1 – G.3 of Appendix G for all sizes of problem structure. 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. The largest computation time is more than ten times as large as the smallest computation time. To stabilize the spread of the 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.4 – G.6 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 table is shown in Tables G.1 – G.3 of Appendix G for small, medium, and large problem structure, respectively. These ANOVA tables are constructed based on the analysis guidelines for split-plot design described in Montgomery (1991).

The ANOVA tables show that the effects of IS and TS are significant to the log-transformed computation time for small, medium and large problem structures. The interaction between IS and TS is significant only for the small problem structure. Due to the interaction effect, the comparisons between two levels of IS should be made within a fixed level of TS as shown in Table G.4 of Appendix G for small problem structure.

For medium and large problem structures, the comparisons between two levels of IS are performed differently from small problem structure. Since the interaction between IS and TS is not significant in medium and large problem structures, the comparisons between two levels of IS can be made over all levels of TS as shown in Table G.5 of Appendix G. The multiple comparisons are based on Duncan's multiple range test. The box plots, analysis of variance, and Duncan's multiple range tests are performed on STATGRAPHICS Plus 3.0.

7.4. Discussion

Recall that in order to identify the level of TS that perform significantly better than the others, the Wilcoxon signed-rank tests are applied on the TWT between the levels of TS. To help visualize the results of Wilcoxon signed-rank tests, a table is constructed in terms of homogeneous groups. A homogeneous group consists of the levels of TS that are not significantly different. The homogeneous groups in terms of the TWT from small problem structure are shown in Table 7.3. In this table, the averages of

TWT and computation time are taken from Table 7.1 and the levels of TS that have sign "X" within the same group imply that they are not significantly different. Table 7.3 shows that there are three homogeneous groups of TWT for the small problem structure. TS5 is significantly different from TS3, TS6, TS4, and TS1. There is no significant difference between TS5 and TS2. TS2 is significantly different from TS4 and TS1, but not significantly different from TS3 and TS6. Clearly, TS5 and TS2 outperform the other levels of TS. Since TS5 and TS2 are not statistically different, the selection between them has to be based on the numerical difference. As TS5 has smaller average TWT than TS2, TS5 is selected as the tabu-search heuristic for small problem structure.

Table 7.3 Homogeneous groups of TWT for small problem structure

Levels of TS	Average TWT	Average CT	Homogeneous Groups		
TS5	257.22	60.0	X		
TS2	258.38	42.3	X	X	
TS3	259.24	43.5		X	X
TS6	259.24	56.1		X	X
TS4	263.97	18.7			X
TS1	264.34	13.6			X

Note: CT = computation time

As Friedman tests showed that there is no significant difference in the TWT between the levels of IS, the analysis is directed toward the computation time required by each level of IS. Since the interaction between IS and TS is significant in terms of computation time, the multiple comparisons between the levels of IS is made by fixing TS at TS5 as the selected tabu-search based heuristic for small problem structure. The results of these multiple comparisons obtained using Duncan's multiple range tests are shown in Table G.4. The homogeneous groups in terms of the computation time for small problem structure are shown in Table 7.4. The averages of the computation time and the TWT are taken from Table 7.1, and the interpretation of the homogeneous groups is similar to previous explanation. As seen from Table 7.4, each level of IS is significantly different from each other in terms of the computation time. IS4 has the smallest average computation time and is statistically different from the other levels. However, IS4 yields the largest average TWT. As the TWT between the levels of IS was analyzed to be not statistically different from each other, the decision to select the best performer would be based on the computation time. Therefore, IS4 is selected as the method to generate initial solutions for TS5 in small problem structure.

Table 7.4 Homogeneous groups of computation time with TS fixed at TS5 (small problem structure)

Levels of IS	Average CT	Average TWT	Homogeneous Groups			
IS4	45.4	259.20	X			
IS2	56.5	255.53		X		
IS1	64.8	255.53			X	
IS3	73.4	258.60				X

Note: CT = computation time; the average CT and average TWT are obtained with TS fixed at TS5

As an afterthought, in section 6.1 of Chapter 6, IS4 was evaluated to yield the largest average percentage deviation and the smallest computation time within the levels of TS fixed at TS5. The heuristic combination that obtained the smallest average percentage deviation was IS3/TS5. This agrees with the experimental results obtained in this section, i.e. IS3 and IS4 are not statistically different in terms of the TWT, but the average TWT obtained by IS3 is numerically smaller than IS4. Furthermore, TS5 is the one that has the smallest average TWT among the six heuristics.

The homogeneous groups in terms of the TWT for medium problem structure are shown in Table 7.5. TS6, TS3, TS2, and TS5 are significantly different from TS4 and TS1. In other words, TS2, TS3, TS5, and TS6 obtained results that are statistically better than TS1 and TS4. However, there is no significant difference between TS2, TS3, TS5, and TS6. If one has to choose among TS2, TS3, TS5, and TS6, it has to be based on the difference in average TWT. Therefore, TS6, as the heuristic that yields the smallest average TWT, is selected to be the tabu-search based heuristic for medium problem structure.

The selection of the initial solution generation method for medium problem structure is based on the computation time since there is no significant difference evaluated between the levels of IS in terms of solution quality (TWT). As the interaction between IS and TS is not significant in terms of computation time, the comparisons between any two levels of IS can be made over all levels of TS. The comparisons between any two levels of IS are done using Duncan's multiple range test and the detailed results are shown in Table G.5. The results of Duncan's analysis are summarized in Table 7.6, which shows four homogeneous groups. In terms of the computation time, all four levels of IS are significantly different from each other. IS4 is evaluated as the one that requires the shortest computation time as well as having the smallest average TWT. Thus, IS4 is selected as the initial solution generation method for medium problem structure.

Table 7.5 Homogeneous groups of TWT for medium problem structure

Levels of TS	Average TWT	Average CT	Homogeneous Groups	
TS6	767.48	2384	X	
TS3	768.48	2226	X	
TS2	769.70	2228	X	
TS5	769.93	2519	X	
TS4	782.42	804		X
TS1	782.93	781		X

Note: CT = computation time

Table 7.6 Homogeneous groups of computation time for medium problem structure

Levels of IS	Average CT	Average TWT	Homogeneous Groups			
IS4	1355	768.77	X			
IS2	1741	774.33		X		
IS1	1810	772.02			X	
IS3	2389	778.84				X

Note: CT = computation time

The results of this statistical analysis agree with the results of the experimentation on medium problem structures mentioned in section 6.2 of chapter 6. Recall from Table 6.6, IS1/TS2, IS1/TS5, IS2/TS2, and IS2/TS5 are the heuristic combinations that obtained the minimum average percentage deviation of all 24 heuristic combinations. The results of the statistical analysis performed on medium problem structures point to the fact that there is no significance difference between IS1, IS2, IS3, and IS4. For TS, Table 6.6 shows that TS2, TS3, TS5 and TS6 obtained average percentage deviations that are equal to or smaller than TS1 and TS4. This means that the first four levels of TS always give better performance than the last two. This is consistent with the results of the statistical analysis summarized in Table 7.5, which shows that TS2, TS3, TS5, and TS6 are statistically different from TS1 and TS4.

For large problem structure, the homogeneous groups in terms of the TWT are shown in Table 7.7. TS2, TS3, TS5 and TS6 are significantly different from TS1 and TS4. This means that TS2, TS3, TS5 and TS6 obtained solutions that are significantly better than TS1 and TS4. However, there are no significant differences between TS2, TS3, TS5 and TS6. Due to the statistical indifference, the decision to select a level of TS has to be based on the numerical difference. Since TS3 has the smallest average TWT, it is selected as the tabu search-based heuristic for large problem structure.

Table 7.7 Homogeneous groups of TWT for large problem structure

Levels of TS	Average TWT	Average CT	Homogeneous Groups	
TS3	932.80	19617	X	
TS6	933.40	18460	X	
TS2	933.90	20979	X	
TS5	937.20	18738	X	
TS4	954.00	6466		X
TS1	955.80	7716		X

Note: CT = computation time

The decision to select a good initial solution generation method for large problem structure is based on the computation time as there is no significant difference between the levels of IS in terms of solution quality (TWT). The four IS methods are compared using the Duncan's analysis and the detailed results are shown in Table G.5 of Appendix G. Four homogeneous groups are identified as shown in Table 7.8. Similar to small and medium problem structure, IS4 in large problem structure is evaluated as the one taking the shortest computation time. In terms of solution quality, IS4 is only 0.04% larger than IS3, which is the one that yields the smallest average TWT. Furthermore, IS3 requires 60% more computation time than IS4. Based on these reasons, IS4 is selected as the initial solution generation method for large problem structure.

Table 7.8 Homogeneous groups of computation time for large problem structure

Levels of IS	Average CT	Average TWT	Homogeneous Groups			
IS4	11242	937.2	X			
IS1	15803	953.3		X		
IS2	16268	937.4			X	
IS3	18004	936.8				X

Note: CT = computation time

The results of this statistical analysis is in agreement with the results of the experimentation performed on large problem structures mentioned in section 6.2 of chapter 6. Although the way that the problems are constructed in section 6.2 is fairly different from the problems used in the statistical analysis, they cover the same size of problem structures. Recall from Table 6.8, IS2 and IS4 turned out to yield average percentage deviation that is smaller or in some cases zero, than IS1 and IS3. Since the results of the statistical analysis point to no-significant difference between the levels of IS, the selection of IS level is mainly based on numerical difference. For TS, Table 6.8 shows that TS2, TS3, TS5 and TS6 obtained average percentage deviations that are equal to or smaller than TS1 and TS4. This is consistent with the results of the statistical analysis summarized in Table 7.7, which shows that TS2, TS3, TS5, and TS6 are statistically different from TS1 and TS4.

7.4.1. The Influence of Initial Solution Generation Methods to Tabu-Search Based Heuristics

The initial solution generated by each IS for each problem instance is shown in Table E.1 – Table E.3 of Appendix E. Obviously, IS4 always results in generating an initial solution that is better than the other three methods. Starting the tabu search from a good initial solution accelerates the search process. The computation time required by a tabu-search based heuristic that uses IS4 to generate the initial solution is always shorter in comparison to the heuristics that use other IS methods. This is the primary advantage of using IS4 as an initial solution generation method. However, the best initial solution may not always result in the best final solution. For example, in the small problem structure previously discussed, after being used in conjunction with the tabu-search based heuristics, IS4 was evaluated to have the shortest average computation time but its average total weighted tardiness was the highest among the four methods. Although employing IS4 as an initial solution generation method in small problem structure is not advantageous in terms of the quality of final solution, it shows a better performance as the problem size increases. In medium problem structure, IS4 was evaluated as the one having the smallest average total weighted tardiness as well as the shortest computation time in comparison to the other IS methods. In large problem structure, the average total weighted tardiness evaluated for IS4 is the second best after IS3 while maintaining its superiority in computation time. Thus, IS4 is preferable as an initial solution generation method.

The efficiency of IS4 is expected in this research. Two reasons explain the efficiency of IS4. First, IS4 is an initial solution generation method that utilizes composite dispatching rules, which are represented in a priority index function (i.e. the ATC function). Composite dispatching rules consider several jobs and machines attributes simultaneously, which is certainly more preferable than a simple dispatching rule such as EDD. Second, more effort was spent in developing IS4 in comparison to developing the other three methods. The effort spent in developing IS4 includes developing an ATC function that incorporates all aspects of the scheduling problem stated in this research, and identifying the appropriate value to be used in the look-ahead parameters.

7.4.2. The Use of Long-Term Memory in Tabu-Search Based Heuristics

Recall that two types of memory are being used among the six different tabu-search based heuristics. The short-term memory is used in TS1 and TS4, while the long-term memory is used in TS2, TS3, T5 and TS6. The short-term memory search always takes shorter computation time than the long-term memory. With the extra amount of time spent in computing with the long-term memory, one might ask if the application of long-term memory actually improves the solution quality. In order to answer the question, comparisons between the heuristics that use short term and long term memory are performed. The comparisons have to be made within the group of heuristics that uses the same type of tabu list (fixed or variable) so that any recognized differences, if exist, would be attributed to the use of different memory features alone. From among the tabu-search based heuristics that use fixed tabu list, TS1 will be compared to TS2 and TS3. Accordingly, TS4 will be compared to TS5 and TS6 from among the heuristics that use variable tabu list. The comparisons are performed using two types of test: Wilcoxon signed rank test and numerical difference test. Wilcoxon signed-rank test is a statistical test and thus, its results carry more weight than numerical difference test. The results of the numerical difference tests are considered when the results of the Wilcoxon signed-rank tests point to non-significant differences. The results of the comparisons using Wilcoxon signed-rank test are obtained from Table F.1 of Appendix F. The numerical difference test compares the numerical differences between the average TWT of two heuristics. The results of both tests are presented in Table 7.9. The entries in each row can be interpreted as follows:

- A "No" means that the two heuristics are not significantly different.
- A "+" sign means that the second heuristic performs better than the first. A "-" sign means that the first heuristic performs better than the second.

Table 7.9 Comparisons between the use of short-term memory and long-term memory

Test Type	Size of Tabu List	Comparisons	Size of Problem Structure		
			Small	Medium	Large
Wilcoxon Signed-Rank Test	Fixed	TS1 & TS2	+	+	+
		TS1 & TS3	No	+	+
	Variable	TS4 & TS5	+	+	+
		TS4 & TS6	No	+	+
Numerical Difference	Fixed	TS1 & TS2	+	+	+
		TS1 & TS3	+	+	+
	Variable	TS4 & TS5	+	+	+
		TS4 & TS6	+	+	+

From Table 7.9, based on Wilcoxon signed-rank test, only 2 of 12 comparisons appear to be not significant. The two non-significant differences occur in small problem structure. For medium and large problem structure, the use of long-term memory is always proven to be beneficial. All 12 comparisons using numerical difference tests favor the use of long-term memory. Thus, it is concluded that the use of long-term memory significantly improves the quality of solution as the problem size moves from small to large.

Within the application of long-term memory, the use of maximal frequency is compared to the use of minimal frequency. In this case, TS2 and TS5 will be compared to TS3 and TS6. The two types of test mentioned above will be applied. The results of the comparisons are presented in Table 7.10.

Based on Wilcoxon signed-rank test, only 2 out of 12 comparisons show preference for LTM-max over LTM-min. The remaining comparisons by Wilcoxon signed-rank test show that the difference is non-significant. The numerical difference tests give quite a different result from Wilcoxon signed-rank test. Using the numerical difference test, 4 of 12 comparisons prefer LTM-max, while 8 comparisons prefer LTM-min. Further observations on the problem structure reveals that the numerical difference test do not exactly give results contrary to Wilcoxon signed-rank test. The two comparisons based on Wilcoxon signed-rank test that prefer LTM-max are identified in small problem structure. The four comparisons based on numerical difference test that prefer LTM-max are also identified in small problem structure. For medium and large problem structures, the numerical difference test shows preference for LTM-min. Therefore, the conclusion drawn from these tests is that LTM-max gives better performance than LTM-min when applied to small problem structure. For medium and large problem structure, although the results are only based on numerical difference test, there is suggestive evidence that LTM-min has resulted in a smaller total weighted tardiness than LTM-max.

Table 7.10 Comparisons between the use of LTM-max and LTM-min

Test Type	Comparisons	Size of Problem Structure		
		Small	Medium	Large
Wilcoxon Signed-Rank Test	TS2 & TS3	No	No	No
	TS2 & TS6	No	No	No
	TS5 & TS3	-	No	No
	TS5 & TS6	-	No	No
Numerical Difference	TS2 & TS3	-	+	+
	TS2 & TS6	-	+	+
	TS5 & TS3	-	+	+
	TS5 & TS6	-	+	+

7.4.3. The Use of Tabu-List Size in Tabu-Search Based Heuristics

Another feature of tabu search that has been applied in this research is the use of different types of tabu-list size. Two different types of tabu-list size have been used: fixed and variable. The fixed tabu-list size is incorporated in TS1, TS2 and TS3, while the variable size is incorporated in TS4, TS5, and TS6. The comparisons between the use of fixed and variable tabu-list size will be done by comparing TS1, TS2, and TS3 with TS4, TS5 and T6, respectively. However, the comparisons are made only between the heuristics that use the same type of memory functions. The two tests used in the previous section will be applied here. The results are presented in Table 7.11.

Table 7.11 Comparisons between the use of fixed and variable size of tabu list

Test Type	Memory Feature	Comparisons	Size of Problem Structure		
			Small	Medium	Large
Wilcoxon Signed-Rank Test	Short	TS1 & TS4	No	No	No
	Long-max	TS2 & TS5	No	No	No
	Long-min	TS3 & TS6	No	No	No
Numerical Difference	Short	TS1 & TS4	+	+	+
	Long-max	TS2 & TS5	+	-	-
	Long-min	TS3 & TS6	No	+	-

Based on Wilcoxon signed-rank test, none of the comparisons turned out to be significantly different. The numerical difference tests show preference for variable tabu-list size in 5 comparisons, preference for fixed tabu-list size in 3 comparisons, and no preference in one comparison. Although one cannot confidently draw a conclusion that variable tabu-list size is preferred over fixed tabu-list size, there is a slight evidence that the use of variable tabu-list size has resulted in smaller total weighted tardiness than fixed tabu-list size. This is particularly true in smaller size of problem structure. But as the size of problem structure grows, the performance of fixed tabu-list size is increasingly better. The change of performance can be seen in the results of the numerical difference tests for medium and large problem. In Table 7.11, the scenario changes from one of no-preference for any tabu-list size in small problem to one of preference for fixed tabu-list size in medium problem, and then to two preferences for fixed tabu-list size in large problem.

In conclusion, the ATC method is preferred as the initial solution generation method to be used with the tabu-search based heuristic. In applying the tabu-search based heuristic, the use of long-term memory is definitely crucial in obtaining a good final solution. The long-term memory should be employed with maximum-frequency strategy to solve small problem structure, but with minimum-frequency strategy for medium and large problem structure. In addition, variable tabu-list size is preferred for solving smaller problem structure, while fixed tabu-list size is preferred as the size of the problem structures increases.

The possible reasoning for the results stated above can be explained by means of the search space. The search space for the small problem structure is not as wide as medium or large problem structures that an intensification search (LTM-max) is sufficiently powerful to identify near optimal/optimal solutions. The use of variable tabu list size further enhances the performance of the intensification search by providing more flexibility in constraining and releasing the tabu restriction. On the other hand, due to the expansion of the search space of medium and large problem structures, a diversification search (LTM-min) is necessary in order to identify near optimal/optimal solutions. Since the diversification search explores the solutions in a new region, keeping the tabu list to a fixed size enables the search process to gradually explore each solution in the new region and identify solutions of good quality.

8. CONCLUSIONS AND SUGGESTIONS FOR FURTHER RESEARCH

Job scheduling problems on unrelated parallel machines with dynamic machine availability and dynamic job release time has been addressed in this research. Unrelated parallel machines are machines that can perform the same function but have different capacity or capability. Since each machine has different capability, the processing 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 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 the industry practice because on-time delivery is a contributing factor to customer satisfaction. Each job in the scheduling problems considered in this research has a job release time, due date, and weight, which can be viewed as a customer's order placement date, shipment date, and priority, respectively. Some jobs considered in the research problem have to be processed in a split mode. These jobs are referred to as split jobs. The difference between the completion times of the split portions of a job should be within a user-specified margin. The constraints imposed on the completion times of split jobs are supported by the Just-In-Time manufacturing concept where inventory has to be maintained at a very low or zero level.

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. An implicit enumeration method such as the branch-and-bound 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 a concept known as tabu search, is 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. Four different methods are developed to generate the initial solution that can be used by tabu search as a starting point. Two of the initial solution generation methods are developed based on a simple dispatching rule known as Earliest Due Date (EDD). The difference between these two methods is that one of them incorporates the mechanism to ensure that the initial solution is feasible and the other one does not. Another method is based on Least Flexible Job (LFJ) and Least Flexible Machine (LFM) rule. The fourth method is based on a composite dispatching rule called Apparent Tardiness Cost (ATC) rule. The ATC method is adapted from a priority index function that was developed for single machine scheduling problem with static job release time and static machine availability. Since the scheduling problem addressed in this research is for unrelated parallel machines with dynamic job release time and dynamic machine availability, the existing priority index function was revised to incorporate these aspects. An experimental study was conducted to identify the appropriate values for the look-ahead parameters used in the function.

In order to assess the quality of the final solutions obtained from tabu-search based heuristics, twelve small problem instances were generated and solved with the branch-and bound technique embedded in Hyper Lingo 4.0, and the tabu-search based heuristics. Using the branch-and-bound technique, 8 out of the 12 problem instances were solved optimally within the stipulated time limit of 72 hours. The optimal solutions are then compared with the solutions obtained from the tabu-search based heuristics. The heuristics obtain solutions that deviates 5.4% in average from the optimal solutions. One of the heuristics (IS3/TS5) obtained solutions that have average percentage deviation of only 1.18%. Furthermore, each heuristic only needs 10 seconds in average to solve the problems in comparison to Hyper Lingo 4.0 that takes 20 hours in average to identify the optimal solutions. Thus, the tabu-search based heuristics are capable of obtaining solutions of good quality within a much shorter time.

Since the optimal solutions for medium and large problem structures are not attainable, the effectiveness of the tabu-search based heuristics is evaluated differently from small problem structure. Five problem instances of medium size and four problem instances of large size were constructed to have zero total weighted tardiness. Then, the tabu-search based heuristics were applied to each problem instance. 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 this reference point. The results show that the average percentage deviation evaluated over the heuristics is 12.9% for medium problem structure and 6.9% for large problem structure.

A more complete experiment with a broader scope was conducted to assess the performance of the heuristics as the size of problem grows from small to medium to large. A multi-factor experiment with split-plot design was conducted. Each problem instance was treated as a block in the experiment. The design of the experiment included two different factors. The four initial solution generation methods (IS1 – IS4) 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 show that IS4, which refers to the ATC method, is recommended as the initial solution generation method for all problem sizes. The ATC 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. TS5, TS6, and TS3 are recommended as the tabu-search based heuristic for the small, medium and large problems, respectively. The use of long-term memory function is definitely recommended in solving all problem structures. The maximum-frequency strategy is recommended to be used with long-term memory function to solve small problem structure. On the other hand, the minimum-frequency strategy is recommended for medium and large problem structures. The variable tabu-list size is preferred for solving smaller problem structure, but the fixed tabu-list size is preferred as the size of the problems grows larger.

As mentioned before, this research focuses on minimizing the total weighted tardiness, which is tailored toward satisfying customer demand. Since machine utilization or workload is not included in the objective function, it is possible that the schedule obtained cause an unbalanced workload on the machines. Therefore, further research may consider balancing workload on machines in addition to minimizing total weighted tardiness. Balancing machine workload is an important issue since unbalanced workload may become the cause of early tool wear or frequent machine breakdown.

Another important objective is to consider minimizing job earliness and tardiness simultaneously. With this objective, the jobs will be processed and completed close to its due date. This is a very relevant objective in modern industry practice as it is closely related to Just-In-Time manufacturing. Ow and Morton (1989) addressed this objective in their research to solve the single-machine scheduling problem.

In this research, the set up times of all jobs are sequence-independent and assumed to be included in the processing time, which may not necessarily be true in all cases. Thus, this research can be extended to consider sequence-dependent set up time for all jobs. Lee et al. (1997) did the study on single machine scheduling with sequence-dependent set up time to minimize the total weighted tardiness. In their work, the job release time and machine availability are completely static.

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.

BIBLIOGRAPHY

Azizoglu, M., and Kirca, O., 1999, Scheduling jobs on unrelated parallel machines to minimize regular total cost functions. *IIE Transactions*, 31:153-159.

Barnes, J. W., Laguna, M., and Glover, F., 1995, An overview of tabu search approaches to production scheduling problems. *Intelligent Scheduling Systems* (ed. Brown, D. E., and Scherer, W. T.), 101-127.

Bradley, J. V., 1968, *Distribution-Free Statistical Tests* (Englewood Cliffs, NJ: Prentice-Hall).

Carlier, J., 1987, Scheduling jobs with release dates and tails on identical machines to minimize the makespan. *European Journal of Operational Research*, 29:298-306.

Carroll, D. C., 1965, *Heuristic sequencing of jobs with single and multiple components*. PhD Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Centeno, G., and Armacost R. L., 1997, Parallel machine scheduling with release time and machine eligibility restrictions. *Computers and Industrial Engineering*, 33(1):273-276.

Cheng, T. C. E., and Sin, C. C. S., 1990, A state-of-the-art review of parallel-machine scheduling research. *European Journal of Operational Research*, 47:271-292.

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

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

Davis, E., and Jaffe, J. M., 1979, *Algorithms for Scheduling Tasks on Unrelated Processors*. Laboratory for Computer Science, Massachusetts Institute of Technology.

Guinet, A., 1995, Scheduling independent jobs on uniform parallel machines to minimize tardiness criteria. *Journal of Intelligent Manufacturing*, 6:95-103.

Hariri, A. M. A., and Potts, C. N., 1991, Heuristics for Scheduling Unrelated Parallel Machines. *Journal of Computers and Operations Research*, 18(3): 323-331.

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

Hubscher, R., and Glover, F., 1994, Applying tabu search with influential diversification to multiprocessor scheduling. *Journal of Computers and Operations Research*, 21(8):877-884.

Glass, C. A., Potts, C. N., and Shade, P., 1994, Unrelated parallel machine scheduling using local search. *Mathematical and Computer Modeling*, 20(2):41-52.

Glover, F., 1986, Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13(55):533-549.

Glover, F., 1989, Tabu Search – Part I. *ORSA Journal on Computing*, 1(3):190-206.

Glover, F., 1990a, Tabu Search: A Tutorial. *Interfaces*, 20:74-94.

Glover, F., 1990b, Tabu Search – Part II. *ORSA Journal on Computing*, 2(1):4-32.

Hyper LINGO version 4.0, 1998 (Chicago: LINDO Systems, Inc.).

Karim, Y., 1999, *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.

Koulamas, C., 1994, The total tardiness problem: review and extensions. *Operations Research*, 42(6):1025-1041.

Lam, K., and Xing, W., 1997, New trends in parallel machine scheduling. *International Journal of Operations & Production Management*, 17:326-338.

Lee, Y. H., Bhaskaran, K., and Pinedo, M., 1997, A heursitic to minimize the total weighted tardiness with sequence-dependent setups. *IIE Transactions*, 29:45-52.

Lenstra, J.K., Rinnooy Kan, A. H. G., and Brucker, P., 1977, Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:343-362.

Lenstra, J. K., Shmoys, D. B., and Tardos, E., 1987, *Approximation algorithms for scheduling unrelated parallel machines*. Report OS-R8714, Centre for Mathematics and Computer Science, Amsterdam.

Logendran, R., and Sonthinen, A., 1997, A tabu search-based approach for scheduling job-shop type flexible manufacturing systems. *Journal of the Operational Research Society*, 48:264-277.

MATLAB version 4.2, 1984-1994 (Natick, MA: The MathWorks, Inc.).

Montgomery, D. C., 1991, *Design and Analysis of Experiments* (NewYork: Wiley).

Muller, F. M., Franca P. M., and Guidini, C. M., 1995, A diversification strategy for a tabu search heuristic to solve the multiprocessor scheduling problem with sequence dependent setup times. *Proceedings of The 11th ISPE/IEE/IFAC International Conference on CAD/CAM Robotics and Factories of the Future '95*, 217-222.

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

Ow, P. S., and Morton, T. E., 1989, The single machine early / tardy problem. *Management Science*, 35(2):177-191.

Park, M. W., and Kim, Y. D., 1997, Search heuristics for a parallel machine scheduling problem with ready times and due dates. *Computers and Industrial Engineering*, 33(3):793-796.

Piersma, N., and Van Dijk, W., 1996, A local search heuristic for unrelated parallel machine scheduling with efficient neighborhood search. *Mathematical and Computer Modeling*, 24(9):11-19.

Rachamadugu, R. V., and Morton, T. E., 1981, *Myopic heuristics for the single machine weighted tardiness problem.* Working Paper #28-81-82, Graduate School of Industrial Administration, Carnegie-Mellon University.

Schutten, J. M. J., and Leussink, R. A. M., 1996, Parallel machine scheduling with release dates, due dates, and family setup times. *International Journal of Production Economics*, 46/47:119-125.

STATGRAPHICS Plus for Windows version 3.0, 1994-1997 (Statistical Graphics Corp.).

Suresh, V., and Chaudhuri, D., 1994, Minimizing maximum tardiness for unrelated parallel machines. *International Journal of Production Economics*, 34:223-229.

Suresh, V., and Chaudhuri, D., 1996a, Bicriteria scheduling problem for unrelated parallel machines. *Computers and Industrial Engineering*, 30(1):77-82.

Suresh, V., and Chaudhuri, D., 1996b, Scheduling of unrelated parallel machines when machine availability is specified. *Production Planning and Control*, 7(4):393-400.

Vepsalainen, A. P. J., and Morton, T. E., 1987, Priority rules for job shops with weighted tardiness costs. *Management Science*, 33(8):1035-1047.

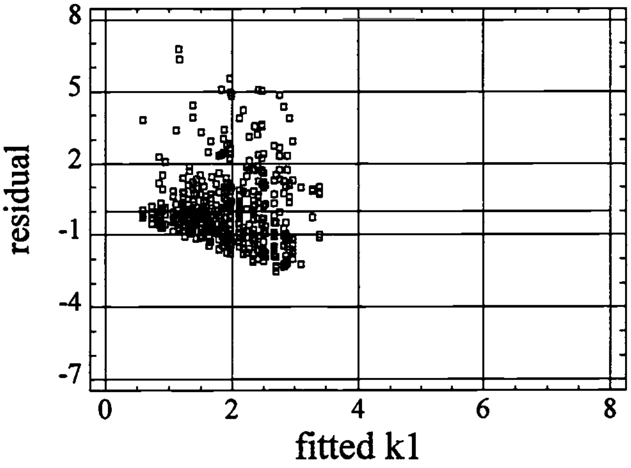
Wilkerson L. J., and Irwin, J. D., 1971, An improved method for scheduling independent tasks. *AIIE Transaction*, 3(3):239-245.

Yaghubian, A. R., Hodgson, T. J., Joines, J. A., Culbreth C. T., and Huang, J. C., 1999, Dry kiln scheduling in furniture production. *IIE Transaction*, 31:733-738.

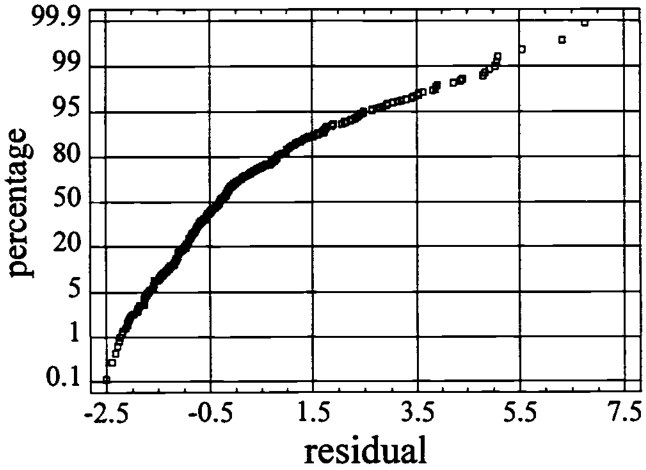
APPENDICES

APPENDIX A. REGRESSION ANALYSIS FOR THE LOOK-AHEAD PARAMETERS

Appendix A.1 Regression Analysis for the First Look-Ahead Parameters (k_1)

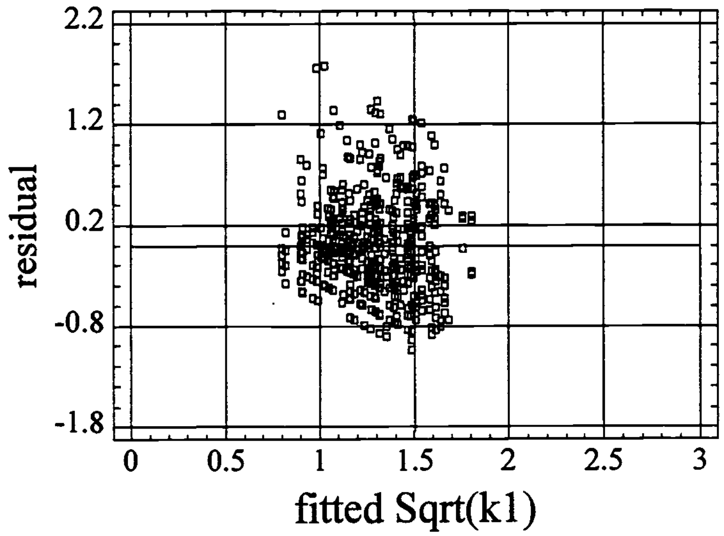


(a) Residual Plot

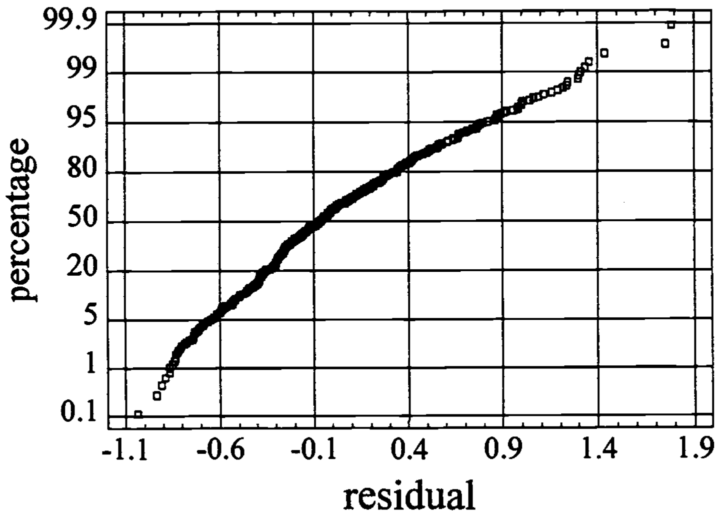


(b) Normal Probability Plot

Figure A.1 Residual Plot and Normal Probability Plot for k_1

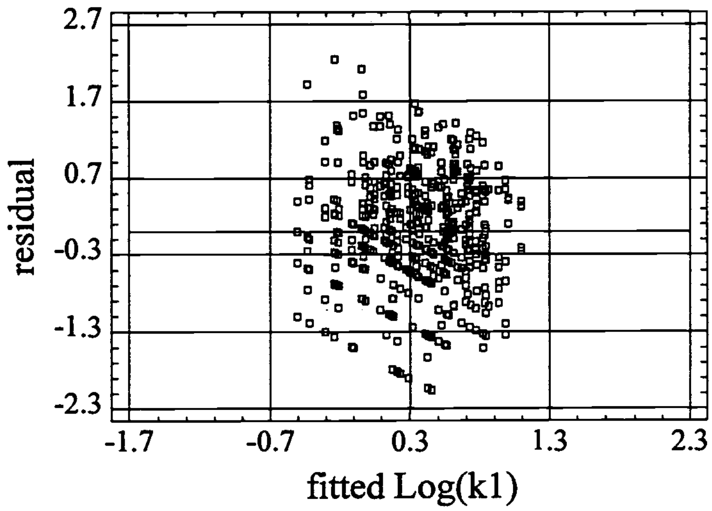


(a) Residual Plot

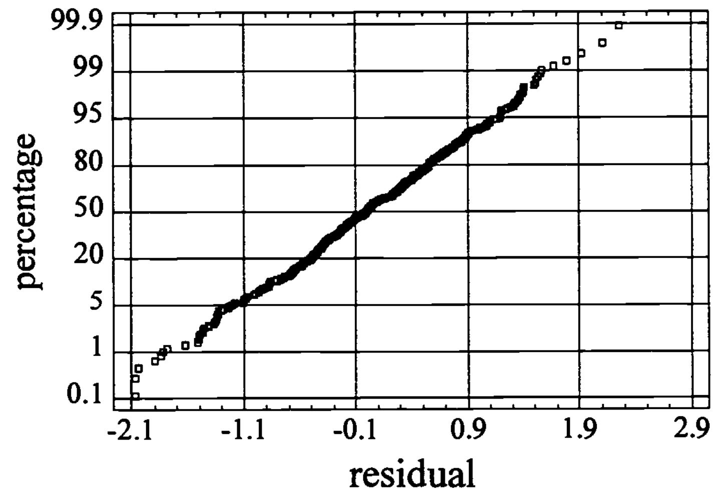


(b) Normal Probability Plot

Figure A.2 Residual Plot and Normal Probability Plot for Sqrt(k_1)



(a) Residual Plot



(b) Normal Probability Plot

Figure A.3 Residual Plot and Normal Probability Plot for Log(k_1)

Table A.1 Analysis of Variance and R^2 statistics for the regression model on k_l

Source	Sum of Squares	Df	Mean Square	F-Ratio	p-value
Model	222.46	6	37.08	17.93	0.0000
Mach	39.71	1	39.71	19.20	0.0000
Range	0.30	1	0.30	0.14	0.7045
Tao	59.10	1	59.10	28.59	0.0000
Mach*Tao	14.72	1	14.72	7.12	0.0079
Range*Tao	9.40	1	9.40	4.55	0.0334
Tao*Tao	60.84	1	60.84	29.43	0.0000
Residual	1137.16	550	2.07		
Total (Corrected)	1359.62	556			

R^2 = 16.3619%
R^2 (adjusted for Df) = 15.4495%

Table A.2 Analysis of Variance and R^2 statistics for the regression model on Sqrt(k_l)

Source	Sum of Squares	Df	Mean Square	F-Ratio	p-value
Model	28.242	6	4.70699	21.22	0.0000
Mach	4.169	1	4.16926	18.79	0.0000
Range	0.102	1	0.10186	0.46	0.4983
Tao	5.620	1	5.62011	25.33	0.0000
Mach*Tao	1.057	1	1.05731	4.77	0.0294
Range*Tao	1.036	1	1.03644	4.67	0.0311
Tao*Tao	5.819	1	5.81897	26.23	0.0000
Residual	122.011	550	0.221838		
Total (Corrected)	150.253	556			

R^2 = 18.7962%
R^2 (adjusted for Df) = 17.9104%

Table A.3 Analysis of Variance and R^2 statistics for the regression model on Log(k_1)

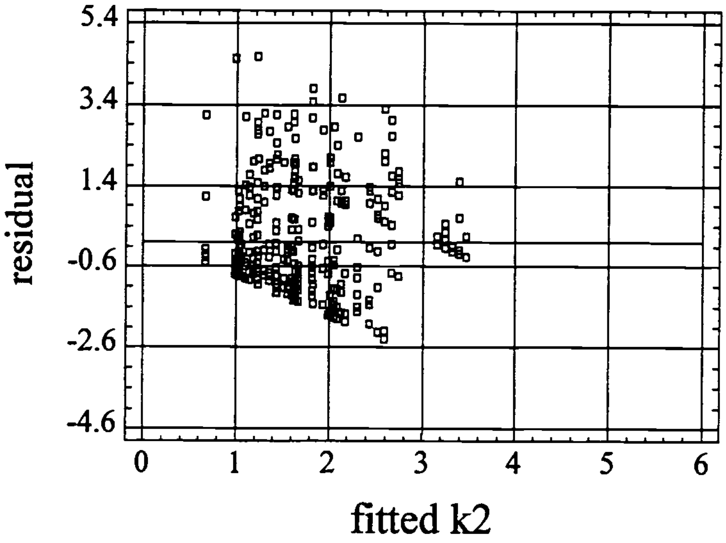
Source	Sum of Squares	Df	Mean Square	F-Ratio	p-value
Model	72.853	5	14.571	27.14	0.0000
Mach	23.742	1	23.742	44.22	0.0000
Range	0.541	1	0.541	1.01	0.3158
Tao	8.195	1	8.195	15.26	0.0001
Range*Tao	2.280	1	2.280	4.25	0.0398
Tao*Tao	9.460	1	9.460	17.62	0.0000
Residual	295.818	551	0.537		
Total (Corrected)	368.671	556			

R^2 = 19.761%
R^2 (adjusted for Df) = 19.0329%

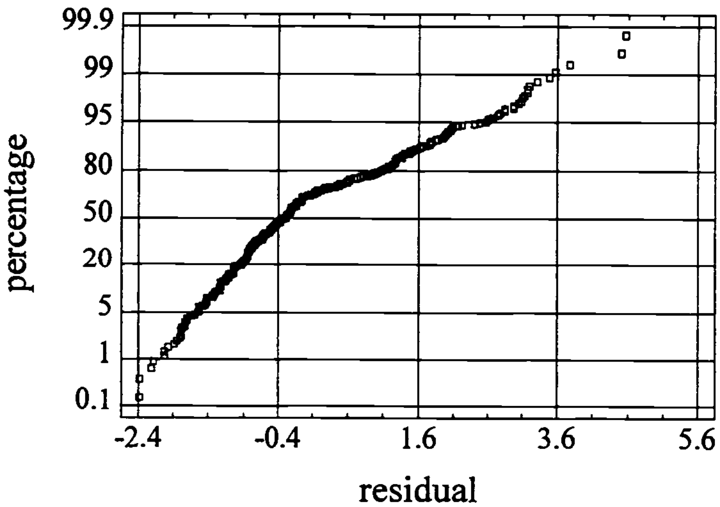
Table A.4 Coefficient Estimates for the regression model on Log(k_1)

Parameter	Estimate	Standard Error
Constant	1.8297	0.2449
Mach	-0.0326	0.0049
Range	-0.2628	0.2618
Tao	-3.4394	0.8803
Range*Tao	-0.9927	0.4817
Tao*Tao	3.4555	0.8232

Appendix A.2 Regression Analysis for The Second Look-Ahead Parameter (k_2)

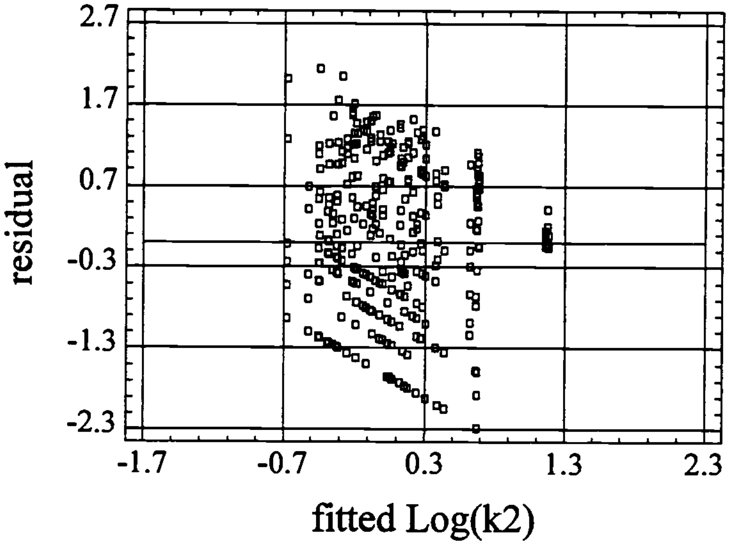


(a) Residual Plot

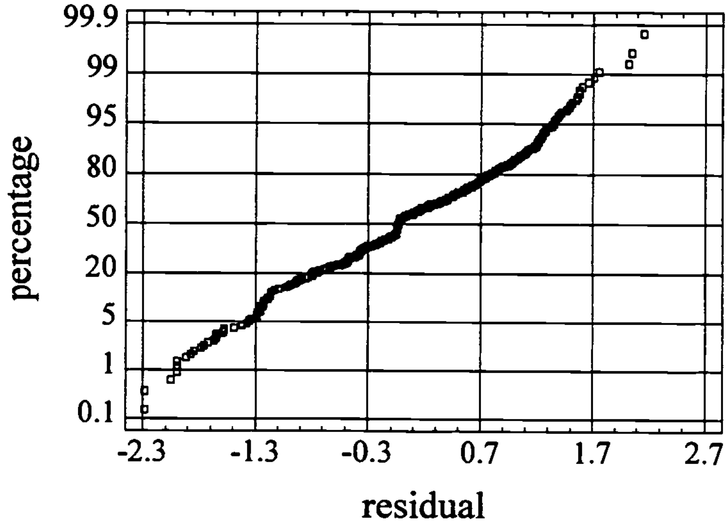


(b) Normal Probability Plot

Figure A.4 Residual Plot and Normal Probability Plot for k_2

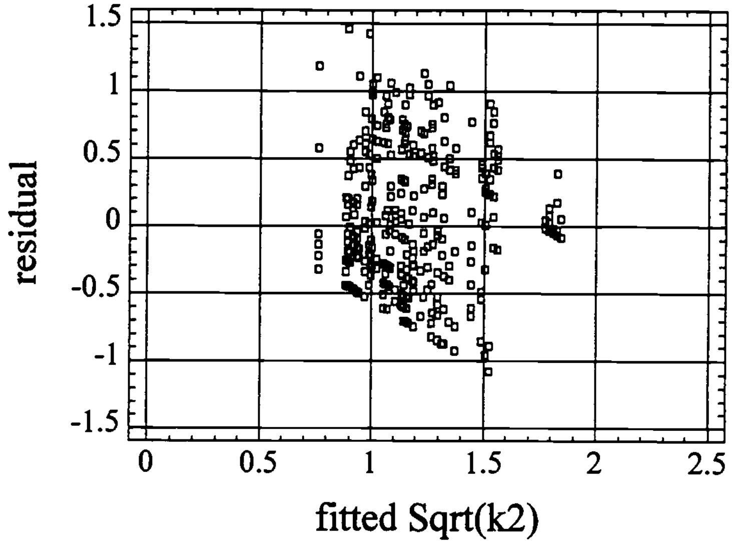


(a) Residual Plot

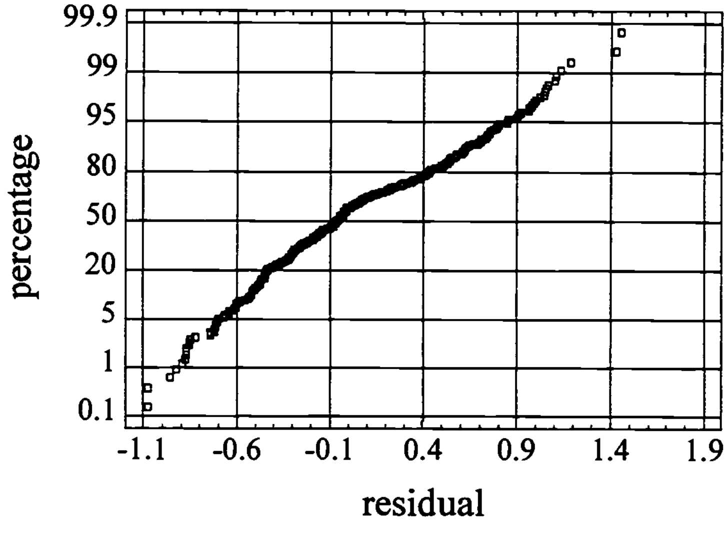


(b) Normal Probability Plot

Figure A.5 Residual Plot and Normal Probability Plot for $Log(k_2)$



(a) Residual Plot



(b) Normal Probability Plot

Figure A.6 Residual Plot and Normal Probability Plot for Sqrt(k_2)

Table A.5 Analysis of Variance and R^2 statistics for the regression model on k_2

Source	Sum of Squares	Df	Mean Square	F-Ratio	p-value
Model	194.632	5	38.926	22.93	0.0000
Job	14.377	1	14.377	8.47	0.0038
Mach	84.383	1	84.383	49.70	0.0000
Tao	5.754	1	5.754	3.39	0.0664
Job*Mach	53.923	1	53.923	31.76	0.0000
Job*Tao	15.225	1	15.225	8.97	0.0029
Residual	660.449	389	1.698		
Total (Corrected)	855.081	394			

R^2 = 22.7618%
R^2 (adjusted for d.f.) = 21.769%

Table A.6 Analysis of Variance and R^2 statistics for the regression model on $Log(k_2)$

Source	Sum of Squares	Df	Mean Square	F-Ratio	p-value
Model	94.051	5	18.810	22.71	0.0000
Job	10.431	1	10.431	12.59	0.0004
Mach	43.318	1	43.318	52.29	0.0000
Tao	1.151	1	1.151	1.39	0.2393
Job*Mach	32.009	1	32.009	38.64	0.0000
Job*Tao	5.880	1	5.880	7.10	0.0080
Residual	322.267	389	0.828		
Total (Corrected)	416.318	394			

R^2= 22.5911%
R^2 (adjusted for d.f.) = 21.5961%

Table A.7 Analysis of Variance and R^2 statistics for the regression model on Sqrt(k_2)

Source	Sum of Squares	Df	Mean Square	F-Ratio	p-value
Model	30.163	5	6.033	24.35	0.0000
Job	2.769	1	2.769	11.18	0.0009
Mach	13.583	1	13.583	54.82	0.0000
Tao	0.598	1	0.598	2.41	0.1212
Job*Mach	9.242	1	9.242	37.30	0.0000
Job*Tao	2.067	1	2.067	8.34	0.0041
Residual	96.376	389	0.248		
Total (Corrected)	126.539	394			

R^2 = 23.8371%
R^2 (adjusted for d.f.) = 22.8582%

Table A.8 Coefficient Estimates for the regression model on Sqrt(k_2)

Parameter	Estimate	Standard Error
Constant	2.2707	0.2169
Job	-0.0174	0.0052
Mach	-0.0912	0.0123
Tao	0.5022	0.3233
Job*Mach	0.0017	0.0003
Job*Tao	-0.0193	0.0067

APPENDIX B. MODEL FORMULATION FOR THE EXAMPLE PROBLEM

Note: This mathematical model is the result of applying the mathematical formulation described in Chapter 4 to the example problem used in Chapter 5. The example problem has 9 jobs, 4 machines and 2 pairs of split jobs. The model is presented in Hyper Lingo 4.0 format.

```
MODEL:
!Objective Function;
MIN = 1*t11 + 1*t21 + 1*t31 + 1*t41 + 2*t12 + 2*t22 + 2*t32 + 2*t42 + 2*t13 + 2*t23
      + 2*t33 + 2*t43 + 3*t14 + 3*t24 + 3*t34 + 3*t44 + 3*t15 + 3*t25 + 3*t35 + 3*t45
      + 2*t16 + 2*t26 + 2*t36 + 2*t46 + 1*t17 + 1*t27 + 1*t37 + 1*t47 + 2*t18 + 2*t28
      + 2*t38 + 2*t48 + 2*t19 + 2*t29 + 2*t39 + 2*t49;

!Constraint (1);
x11 + x21 + x31 + x41 = 1;
x12 + x22 + x32 + x42 = 1;
x13 + x23 + x33 + x43 = 1;
x14 + x24 + x34 + x44 = 1;
x15 + x25 + x35 + x45 = 1;
x16 + x26 + x36 + x46 = 1;
x17 + x27 + x37 + x47 = 1;
x18 + x28 + x38 + x48 = 1;
x19 + x29 + x39 + x49 = 1;

!Constraint (2);
x11 * (1 + 10) <= c11;
x21 * (1 + 1000) <= c21;
x31 * (1 + 1000) <= c31;
x41 * (1 + 1000) <= c41;
x12 * (4 + 4) <= c12;
x22 * (4 + 8) <= c22;
x32 * (4 + 9) <= c32;
x42 * (4 + 9) <= c42;
x13 * (3 + 1000) <= c13;
x23 * (3 + 5) <= c23;
x33 * (3 + 8) <= c33;
x43 * (3 + 8) <= c43;
x14 * (4 + 7) <= c14;
x24 * (4 + 9) <= c24;
x34 * (4 + 1000) <= c34;
x44 * (4 + 1000) <= c44;
x15 * (4 + 8) <= c15;
x25 * (4 + 11) <= c25;
x35 * (4 + 1000) <= c35;
x45 * (4 + 1000) <= c45;
x16 * (9 + 1000) <= c16;
x26 * (9 + 4) <= c26;
x36 * (9 + 6) <= c36;
x46 * (9 + 6) <= c46;
x17 * (8 + 8) <= c17;
x27 * (8 + 10) <= c27;
x37 * (8 + 1000) <= c37;
x47 * (8 + 1000) <= c47;
x18 * (5 + 1000) <= c18;
x28 * (5 + 6) <= c28;
x38 * (5 + 9) <= c38;
x48 * (5 + 9) <= c48;
x19 * (5 + 1000) <= c19;
x29 * (5 + 7) <= c29;
x39 * (5 + 11) <= c39;
x49 * (5 + 11) <= c49;

!Constraints (3);
x11 * (0 + 10) <= c11;
x21 * (2 + 1000) <= c21;
x31 * (2 + 1000) <= c31;
x41 * (5 + 1000) <= c41;
x12 * (0 + 4) <= c12;
x22 * (2 + 8) <= c22;
x32 * (2 + 9) <= c32;
x42 * (5 + 9) <= c42;
x13 * (0 + 1000) <= c13;
x23 * (2 + 5) <= c23;
x33 * (2 + 8) <= c33;
x43 * (5 + 8) <= c43;
x14 * (0 + 7) <= c14;
x24 * (2 + 9) <= c24;
x34 * (2 + 1000) <= c34;
x44 * (5 + 1000) <= c44;
x15 * (0 + 8) <= c15;
x25 * (2 + 11) <= c25;
x35 * (2 + 1000) <= c35;
x45 * (5 + 1000) <= c45;
x16 * (0 + 1000) <= c16;
x26 * (2 + 4) <= c26;
x36 * (2 + 6) <= c36;
x46 * (5 + 6) <= c46;
x17 * (0 + 8) <= c17;
x27 * (2 + 10) <= c27;
x37 * (2 + 1000) <= c37;
x47 * (5 + 1000) <= c47;
x18 * (0 + 1000) <= c18;
x28 * (2 + 6) <= c28;
x38 * (2 + 9) <= c38;
x48 * (5 + 9) <= c48;
x19 * (0 + 1000) <= c19;
x29 * (2 + 7) <= c29;
x39 * (2 + 11) <= c39;
x49 * (5 + 11) <= c49;

!Constraints (4);
c11 <= 10000 * x11;
c21 <= 10000 * x21;
c31 <= 10000 * x31;
c41 <= 10000 * x41;
c12 <= 10000 * x12;
c22 <= 10000 * x22;
c32 <= 10000 * x32;
c42 <= 10000 * x42;
c13 <= 10000 * x13;
c23 <= 10000 * x23;
c33 <= 10000 * x33;
c43 <= 10000 * x43;
```

```
c14 <= 10000 * x14;
c24 <= 10000 * x24;
c34 <= 10000 * x34;
c44 <= 10000 * x44;
c15 <= 10000 * x15;
c25 <= 10000 * x25;
c35 <= 10000 * x35;
c45 <= 10000 * x45;
c16 <= 10000 * x16;
c26 <= 10000 * x26;
c36 <= 10000 * x36;
c46 <= 10000 * x46;
c17 <= 10000 * x17;
c27 <= 10000 * x27;
c37 <= 10000 * x37;
c47 <= 10000 * x47;
c18 <= 10000 * x18;
c28 <= 10000 * x28;
c38 <= 10000 * x38;
c48 <= 10000 * x48;
c19 <= 10000 * x19;
c29 <= 10000 * x29;
c39 <= 10000 * x39;
c49 <= 10000 * x49;

!Constraints (5);
c12 – c11 + 10000*(1 – y112) >= x12*4;
c13 – c11 + 10000*(1 – y113) >= x13*1000;
c14 – c11 + 10000*(1 – y114) >= x14*7;
c15 – c11 + 10000*(1 – y115) >= x15*8;
c16 – c11 + 10000*(1 – y116) >= x16*1000;
c17 – c11 + 10000*(1 – y117) >= x17*8;
c18 – c11 + 10000*(1 – y118) >= x18*1000;
c19 – c11 + 10000*(1 – y119) >= x19*1000;
c13 – c12 + 10000*(1 – y123) >= x13*1000;
c14 – c12 + 10000*(1 – y124) >= x14*7;
c15 – c12 + 10000*(1 – y125) >= x15*8;
c16 – c12 + 10000*(1 – y126) >= x16*1000;
c17 – c12 + 10000*(1 – y127) >= x17*8;
c18 – c12 + 10000*(1 – y128) >= x18*1000;
c19 – c12 + 10000*(1 – y129) >= x19*1000;
c14 – c13 + 10000*(1 – y134) >= x14*7;
c15 – c13 + 10000*(1 – y135) >= x15*8;
c16 – c13 + 10000*(1 – y136) >= x16*1000;
c17 – c13 + 10000*(1 – y137) >= x17*8;
c18 – c13 + 10000*(1 – y138) >= x18*1000;
c19 – c13 + 10000*(1 – y139) >= x19*1000;
c15 – c14 + 10000*(1 – y145) >= x15*8;
c16 – c14 + 10000*(1 – y146) >= x16*1000;
c17 – c14 + 10000*(1 – y147) >= x17*8;
c18 – c14 + 10000*(1 – y148) >= x18*1000;
c19 – c14 + 10000*(1 – y149) >= x19*1000;
c16 – c15 + 10000*(1 – y156) >= x16*1000;
c17 – c15 + 10000*(1 – y157) >= x17*8;
c18 – c15 + 10000*(1 – y158) >= x18*1000;
c19 – c15 + 10000*(1 – y159) >= x19*1000;
c17 – c16 + 10000*(1 – y167) >= x17*8;
c18 – c16 + 10000*(1 – y168) >= x18*1000;
c19 – c16 + 10000*(1 – y169) >= x19*1000;
c18 – c17 + 10000*(1 – y178) >= x18*1000;
c19 – c17 + 10000*(1 – y179) >= x19*1000;
c19 – c18 + 10000*(1 – y189) >= x19*1000;
c22 – c21 + 10000*(1 – y212) >= x22*8;
c23 – c21 + 10000*(1 – y213) >= x23*5;
c24 – c21 + 10000*(1 – y214) >= x24*9;
c25 – c21 + 10000*(1 – y215) >= x25*11;
c26 – c21 + 10000*(1 – y216) >= x26*4;
c27 – c21 + 10000*(1 – y217) >= x27*10;
c28 – c21 + 10000*(1 – y218) >= x28*6;
c29 – c21 + 10000*(1 – y219) >= x29*7;
c23 – c22 + 10000*(1 – y223) >= x23*5;
c24 – c22 + 10000*(1 – y224) >= x24*9;
c25 – c22 + 10000*(1 – y225) >= x25*11;
c26 – c22 + 10000*(1 – y226) >= x26*4;
c27 – c22 + 10000*(1 – y227) >= x27*10;
c28 – c22 + 10000*(1 – y228) >= x28*6;
c29 – c22 + 10000*(1 – y229) >= x29*7;
c24 – c23 + 10000*(1 – y234) >= x24*9;
c25 – c23 + 10000*(1 – y235) >= x25*11;
c26 – c23 + 10000*(1 – y236) >= x26*4;
c27 – c23 + 10000*(1 – y237) >= x27*10;
c28 – c23 + 10000*(1 – y238) >= x28*6;
c29 – c23 + 10000*(1 – y239) >= x29*7;
c25 – c24 + 10000*(1 – y245) >= x25*11;
c26 – c24 + 10000*(1 – y246) >= x26*4;
c27 – c24 + 10000*(1 – y247) >= x27*10;
c28 – c24 + 10000*(1 – y248) >= x28*6;
c29 – c24 + 10000*(1 – y249) >= x29*7;
c26 – c25 + 10000*(1 – y256) >= x26*4;
c27 – c25 + 10000*(1 – y257) >= x27*10;
c28 – c25 + 10000*(1 – y258) >= x28*6;
c29 – c25 + 10000*(1 – y259) >= x29*7;
c27 – c26 + 10000*(1 – y267) >= x27*10;
c28 – c26 + 10000*(1 – y268) >= x28*6;
c29 – c26 + 10000*(1 – y269) >= x29*7;
c28 – c27 + 10000*(1 – y278) >= x28*6;
c29 – c27 + 10000*(1 – y279) >= x29*7;
c29 – c28 + 10000*(1 – y289) >= x29*7;
c32 – c31 + 10000*(1 – y312) >= x32*9;
c33 – c31 + 10000*(1 – y313) >= x33*8;
c34 – c31 + 10000*(1 – y314) >= x34*1000;
c35 – c31 + 10000*(1 – y315) >= x35*1000;
c36 – c31 + 10000*(1 – y316) >= x36*6;
c37 – c31 + 10000*(1 – y317) >= x37*1000;
c38 – c31 + 10000*(1 – y318) >= x38*9;
c39 – c31 + 10000*(1 – y319) >= x39*11;
c33 – c32 + 10000*(1 – y323) >= x33*8;
c34 – c32 + 10000*(1 – y324) >= x34*1000;
c35 – c32 + 10000*(1 – y325) >= x35*1000;
c36 – c32 + 10000*(1 – y326) >= x36*6;
c37 – c32 + 10000*(1 – y327) >= x37*1000;
c38 – c32 + 10000*(1 – y328) >= x38*9;
c39 – c32 + 10000*(1 – y329) >= x39*11;
c34 – c33 + 10000*(1 – y334) >= x34*1000;
c35 – c33 + 10000*(1 – y335) >= x35*1000;
c36 – c33 + 10000*(1 – y336) >= x36*6;
c37 – c33 + 10000*(1 – y337) >= x37*1000;
c38 – c33 + 10000*(1 – y338) >= x38*9;
```

```
c39 – c33 + 10000*(1 – y339) >= x39*11;
c35 – c34 + 10000*(1 – y345) >= x35*1000;
c36 – c34 + 10000*(1 – y346) >= x36*6;
c37 – c34 + 10000*(1 – y347) >= x37*1000;
c38 – c34 + 10000*(1 – y348) >= x38*9;
c39 – c34 + 10000*(1 – y349) >= x39*11;
c36 – c35 + 10000*(1 – y356) >= x36*6;
c37 – c35 + 10000*(1 – y357) >= x37*1000;
c38 – c35 + 10000*(1 – y358) >= x38*9;
c39 – c35 + 10000*(1 – y359) >= x39*11;
c37 – c36 + 10000*(1 – y367) >= x37*1000;
c38 – c36 + 10000*(1 – y368) >= x38*9;
c39 – c36 + 10000*(1 – y369) >= x39*11;
c38 – c37 + 10000*(1 – y378) >= x38*9;
c39 – c37 + 10000*(1 – y379) >= x39*11;
c39 – c38 + 10000*(1 – y389) >= x39*11;
c42 – c41 + 10000 * (1 – y412) >= x42 * 9;
c43 – c41 + 10000*(1 – y413) >= x43 * 8;
c44 – c41 + 10000*(1 – y414) >= x44*1000;
c45 – c41 + 10000*(1 – y415) >= x45*1000;
c46 – c41 + 10000*(1 – y416) >= x46*6;
c47 – c41 + 10000*(1 – y417) >= x47*1000;
c48 – c41 + 10000*(1 – y418) >= x48*9;
c49 – c41 + 10000*(1 – y419) >= x49*11;
c43 – c42 + 10000*(1 – y423) >= x43*8;
c44 – c42 + 10000*(1 – y424) >= x44*1000;
c45 – c42 + 10000*(1 – y425) >= x45*1000;
c46 – c42 + 10000*(1 – y426) >= x46*6;
c47 – c42 + 10000*(1 – y427) >= x47*1000;
c48 – c42 + 10000*(1 – y428) >= x48*9;
c49 – c42 + 10000*(1 – y429) >= x49*11;
c44 – c43 + 10000*(1 – y434) >= x44*1000;
c45 – c43 + 10000*(1 – y435) >= x45*1000;
c46 – c43 + 10000*(1 – y436) >= x46*6;
c47 – c43 + 10000*(1 – y437) >= x47*1000;
c48 – c43 + 10000*(1 – y438) >= x48*9;
c49 – c43 + 10000*(1 – y439) >= x49*11;
c45 – c44 + 10000*(1 – y445) >= x45*1000;
c46 – c44 + 10000*(1 – y446) >= x46*6;
c47 – c44 + 10000*(1 – y447) >= x47*1000;
c48 – c44 + 10000*(1 – y448) >= x48*9;
c49 – c44 + 10000*(1 – y449) >= x49*11;
c46 – c45 + 10000*(1 – y456) >= x46*6;
c47 – c45 + 10000*(1 – y457) >= x47*1000;
c48 – c45 + 10000*(1 – y458) >= x48*9;
c49 – c45 + 10000*(1 – y459) >= x49*11;
c47 – c46 + 10000*(1 – y467) >= x47*1000;
c48 – c46 + 10000*(1 – y468) >= x48*9;
c49 – c46 + 10000*(1 – y469) >= x49*11;
c48 – c47 + 10000*(1 – y478) >= x48*9;
c49 – c47 + 10000*(1 – y479) >= x49*11;
c49 – c48 + 10000*(1 – y489) >= x49*11;

!Constraints (6);
c11 – c12 + 10000 * y112 >= x11 * 10;
c11 – c13 + 10000 * y113 >= x11 * 10;
c11 – c14 + 10000 * y114 >= x11 * 10;
c11 – c15 + 10000 * y115 >= x11 * 10;
c11 – c16 + 10000 * y116 >= x11 * 10;
c11 – c17 + 10000 * y117 >= x11 * 10;
c11 – c18 + 10000 * y118 >= x11 * 10;
c11 – c19 + 10000 * y119 >= x11 * 10;
c12 – c13 + 10000 * y123 >= x12 * 4;
c12 – c14 + 10000 * y124 >= x12 * 4;
c12 – c15 + 10000 * y125 >= x12 * 4;
c12 – c16 + 10000 * y126 >= x12 * 4;
c12 – c17 + 10000 * y127 >= x12 * 4;
c12 – c18 + 10000 * y128 >= x12 * 4;
c12 – c19 + 10000 * y129 >= x12 * 4;
c13 – c14 + 10000 * y134 >= x13 * 1000;
c13 – c15 + 10000 * y135 >= x13 * 1000;
c13 – c16 + 10000 * y136 >= x13 * 1000;
c13 – c17 + 10000 * y137 >= x13 * 1000;
c13 – c18 + 10000 * y138 >= x13 * 1000;
c13 – c19 + 10000 * y139 >= x13 * 1000;
c14 – c15 + 10000 * y145 >= x14 * 7;
c14 – c16 + 10000 * y146 >= x14 * 7;
c14 – c17 + 10000 * y147 >= x14 * 7;
c14 – c18 + 10000 * y148 >= x14 * 7;
c14 – c19 + 10000 * y149 >= x14 * 7;
c15 – c16 + 10000 * y156 >= x15 * 8;
c15 – c17 + 10000 * y157 >= x15 * 8;
c15 – c18 + 10000 * y158 >= x15 * 8;
c15 – c19 + 10000 * y159 >= x15 * 8;
c16 – c17 + 10000 * y167 >= x16 * 1000;
c16 – c18 + 10000 * y168 >= x16 * 1000;
c16 – c19 + 10000 * y169 >= x16 * 1000;
c17 – c18 + 10000 * y178 >= x17 * 8;
c17 – c19 + 10000 * y179 >= x17 * 8;
c18 – c19 + 10000 * y189 >= x18 * 1000;
c21 – c22 + 10000 * y212 >= x21 * 1000;
c21 – c23 + 10000 * y213 >= x21 * 1000;
c21 – c24 + 10000 * y214 >= x21 * 1000;
c21 – c25 + 10000 * y215 >= x21 * 1000;
c21 – c26 + 10000 * y216 >= x21 * 1000;
c21 – c27 + 10000 * y217 >= x21 * 1000;
c21 – c28 + 10000 * y218 >= x21 * 1000;
c21 – c29 + 10000 * y219 >= x21 * 1000;
c22 – c23 + 10000 * y223 >= x22 * 8;
c22 – c24 + 10000 * y224 >= x22 * 8;
c22 – c25 + 10000 * y225 >= x22 * 8;
c22 – c26 + 10000 * y226 >= x22 * 8;
c22 – c27 + 10000 * y227 >= x22 * 8;
c22 – c28 + 10000 * y228 >= x22 * 8;
c22 – c29 + 10000 * y229 >= x22 * 8;
c23 – c24 + 10000 * y234 >= x23 * 5;
c23 – c25 + 10000 * y235 >= x23 * 5;
c23 – c26 + 10000 * y236 >= x23 * 5;
c23 – c27 + 10000 * y237 >= x23 * 5;
c23 – c28 + 10000 * y238 >= x23 * 5;
```

```
c23 – c29 + 10000 * y239 >= x23 * 5;
c24 – c25 + 10000 * y245 >= x24 * 9;
c24 – c26 + 10000 * y246 >= x24 * 9;
c24 – c27 + 10000 * y247 >= x24 * 9;
c24 – c28 + 10000 * y248 >= x24 * 9;
c24 – c29 + 10000 * y249 >= x24 * 9;
c25 – c26 + 10000 * y256 >= x25 * 11;
c25 – c27 + 10000 * y257 >= x25 * 11;
c25 – c28 + 10000 * y258 >= x25 * 11;
c25 – c29 + 10000 * y259 >= x25 * 11;
c26 – c27 + 10000 * y267 >= x26 * 4;
c26 – c28 + 10000 * y268 >= x26 * 4;
c26 – c29 + 10000 * y269 >= x26 * 4;
c27 – c28 + 10000 * y278 >= x27 * 10;
c27 – c29 + 10000 * y279 >= x27 * 10;
c28 – c29 + 10000 * y289 >= x28 * 6;
c31 – c32 + 10000 * y312 >= x31 * 1000;
c31 – c33 + 10000 * y313 >= x31 * 1000;
c31 – c34 + 10000 * y314 >= x31 * 1000;
c31 – c35 + 10000 * y315 >= x31 * 1000;
c31 – c36 + 10000 * y316 >= x31 * 1000;
c31 – c37 + 10000 * y317 >= x31 * 1000;
c31 – c38 + 10000 * y318 >= x31 * 1000;
c31 – c39 + 10000 * y319 >= x31 * 1000;
c32 – c33 + 10000 * y323 >= x32 * 9;
c32 – c34 + 10000 * y324 >= x32 * 9;
c32 – c35 + 10000 * y325 >= x32 * 9;
c32 – c36 + 10000 * y326 >= x32 * 9;
c32 – c37 + 10000 * y327 >= x32 * 9;
c32 – c38 + 10000 * y328 >= x32 * 9;
c32 – c39 + 10000 * y329 >= x32 * 9;
c33 – c34 + 10000 * y334 >= x33 * 8;
c33 – c35 + 10000 * y335 >= x33 * 8;
c33 – c36 + 10000 * y336 >= x33 * 8;
c33 – c37 + 10000 * y337 >= x33 * 8;
c33 – c38 + 10000 * y338 >= x33 * 8;
c33 – c39 + 10000 * y339 >= x33 * 8;
c34 – c35 + 10000 * y345 >= x34 * 1000;
c34 – c36 + 10000 * y346 >= x34 * 1000;
c34 – c37 + 10000 * y347 >= x34 * 1000;
c34 – c38 + 10000 * y348 >= x34 * 1000;
c34 – c39 + 10000 * y349 >= x34 * 1000;
c35 – c36 + 10000 * y356 >= x35 * 1000;
c35 – c37 + 10000 * y357 >= x35 * 1000;
c35 – c38 + 10000 * y358 >= x35 * 1000;
c35 – c39 + 10000 * y359 >= x35 * 1000;
c36 – c37 + 10000 * y367 >= x36 * 6;
c36 – c38 + 10000 * y368 >= x36 * 6;
c36 – c39 + 10000 * y369 >= x36 * 6;
c37 – c38 + 10000 * y378 >= x37 * 1000;
c37 – c39 + 10000 * y379 >= x37 * 1000;
c38 – c39 + 10000 * y389 >= x38 * 9;
c41 – c42 + 10000 * y412 >= x41 * 1000;
c41 – c43 + 10000 * y413 >= x41 * 1000;
c41 – c44 + 10000 * y414 >= x41 * 1000;
c41 – c45 + 10000 * y415 >= x41 * 1000;
c41 – c46 + 10000 * y416 >= x41 * 1000;
c41 – c47 + 10000 * y417 >= x41 * 1000;
c41 – c48 + 10000 * y418 >= x41 * 1000;
c41 – c49 + 10000 * y419 >= x41 * 1000;
c42 – c43 + 10000 * y423 >= x42 * 9;
c42 – c44 + 10000 * y424 >= x42 * 9;
c42 – c45 + 10000 * y425 >= x42 * 9;
c42 – c46 + 10000 * y426 >= x42 * 9;
c42 – c47 + 10000 * y427 >= x42 * 9;
c42 – c48 + 10000 * y428 >= x42 * 9;
c42 – c49 + 10000 * y429 >= x42 * 9;
c43 – c44 + 10000 * y434 >= x43 * 8;
c43 – c45 + 10000 * y435 >= x43 * 8;
c43 – c46 + 10000 * y436 >= x43 * 8;
c43 – c47 + 10000 * y437 >= x43 * 8;
c43 – c48 + 10000 * y438 >= x43 * 8;
c43 – c49 + 10000 * y439 >= x43 * 8;
c44 – c45 + 10000 * y445 >= x44 * 1000;
c44 – c46 + 10000 * y446 >= x44 * 1000;
c44 – c47 + 10000 * y447 >= x44 * 1000;
c44 – c48 + 10000 * y448 >= x44 * 1000;
c44 – c49 + 10000 * y449 >= x44 * 1000;
c45 – c46 + 10000 * y456 >= x45 * 1000;
c45 – c47 + 10000 * y457 >= x45 * 1000;
c45 – c48 + 10000 * y458 >= x45 * 1000;
c45 – c49 + 10000 * y459 >= x45 * 1000;
c46 – c47 + 10000 * y467 >= x46 * 6;
c46 – c48 + 10000 * y468 >= x46 * 6;
c46 – c49 + 10000 * y469 >= x46 * 6;
c47 – c48 + 10000 * y478 >= x47 * 1000;
c47 – c49 + 10000 * y479 >= x47 * 1000;
c48 – c49 + 10000 * y489 >= x48 * 9;

!Constraints (7)
c14 – c15 <= 1 + 10000*(2 – x14 – x15);
c14 – c25 <= 1 + 10000*(2 – x14 – x25);
c14 – c35 <= 1 + 10000*(2 – x14 – x35);
c14 – c45 <= 1 + 10000*(2 – x14 – x45);
c24 – c15 <= 1 + 10000*(2 – x24 – x15);
c24 – c25 <= 1 + 10000*(2 – x24 – x25);
c24 – c35 <= 1 + 10000*(2 – x24 – x35);
c24 – c45 <= 1 + 10000*(2 – x24 – x45);
c34 – c15 <= 1 + 10000*(2 – x34 – x15);
c34 – c25 <= 1 + 10000*(2 – x34 – x25);
c34 – c35 <= 1 + 10000*(2 – x34 – x35);
c34 – c45 <= 1 + 10000*(2 – x34 – x45);
c44 – c15 <= 1 + 10000*(2 – x44 – x15);
c44 – c25 <= 1 + 10000*(2 – x44 – x25);
c44 – c35 <= 1 + 10000*(2 – x44 – x35);
c44 – c45 <= 1 + 10000*(2 – x44 – x45);
c18 – c19 <= 1 + 10000*(2 – x18 – x19);
c18 – c29 <= 1 + 10000*(2 – x18 – x29);
c18 – c39 <= 1 + 10000*(2 – x18 – x39);
c18 – c49 <= 1 + 10000*(2 – x18 – x49);
```

```
c28 – c19 <= 1 + 10000*(2 – x28 – x19);
c28 – c29 <= 1 + 10000*(2 – x28 – x29);
c28 – c39 <= 1 + 10000*(2 – x28 – x39);
c28 – c49 <= 1 + 10000*(2 – x28 – x49);
c38 – c19 <= 1 + 10000*(2 – x38 – x19);
c38 – c29 <= 1 + 10000*(2 – x38 – x29);
c38 – c39 <= 1 + 10000*(2 – x38 – x39);
c38 – c49 <= 1 + 10000*(2 – x38 – x49);
c48 – c19 <= 1 + 10000*(2 – x48 – x19);
c48 – c29 <= 1 + 10000*(2 – x48 – x29);
c48 – c39 <= 1 + 10000*(2 – x48 – x39);
c48 – c49 <= 1 + 10000*(2 – x48 – x49);

!Constraints (8);
c15 – c14 <= 1 + 10000*(2 – x14 – x15);
c25 – c14 <= 1 + 10000*(2 – x14 – x25);
c35 – c14 <= 1 + 10000*(2 – x14 – x35);
c45 – c14 <= 1 + 10000*(2 – x14 – x45);
c15 – c24 <= 1 + 10000*(2 – x24 – x15);
c25 – c24 <= 1 + 10000*(2 – x24 – x25);
c35 – c24 <= 1 + 10000*(2 – x24 – x35);
c45 – c24 <= 1 + 10000*(2 – x24 – x45);
c15 – c34 <= 1 + 10000*(2 – x34 – x15);
c25 – c34 <= 1 + 10000*(2 – x34 – x25);
c35 – c34 <= 1 + 10000*(2 – x34 – x35);
c45 – c34 <= 1 + 10000*(2 – x34 – x45);
c15 – c44 <= 1 + 10000*(2 – x44 – x15);
c25 – c44 <= 1 + 10000*(2 – x44 – x25);
c35 – c44 <= 1 + 10000*(2 – x44 – x35);
c45 – c44 <= 1 + 10000*(2 – x44 – x45);
c19 – c18 <= 1 + 10000*(2 – x18 – x19);
c29 – c18 <= 1 + 10000*(2 – x18 – x29);
c39 – c18 <= 1 + 10000*(2 – x18 – x39);
c49 – c18 <= 1 + 10000*(2 – x18 – x49);
c19 – c28 <= 1 + 10000*(2 – x28 – x19);
c29 – c28 <= 1 + 10000*(2 – x28 – x29);
c39 – c28 <= 1 + 10000*(2 – x28 – x39);
c49 – c28 <= 1 + 10000*(2 – x28 – x49);
c19 – c38 <= 1 + 10000*(2 – x38 – x19);
c29 – c38 <= 1 + 10000*(2 – x38 – x29);
c39 – c38 <= 1 + 10000*(2 – x38 – x39);
c49 – c38 <= 1 + 10000*(2 – x38 – x49);
c19 – c48 <= 1 + 10000*(2 – x48 – x19);
c29 – c48 <= 1 + 10000*(2 – x48 – x29);
c39 – c48 <= 1 + 10000*(2 – x48 – x39);
c49 – c48 <= 1 + 10000*(2 – x48 – x49);

!Constraints (9);
c11 – 15 <= t11;
c21 – 15 <= t21;
c31 – 15 <= t31;
c41 – 15 <= t41;
c12 – 12 <= t12;
c22 – 12 <= t22;
c32 – 12 <= t32;
c42 – 12 <= t42;
c13 – 7 <= t13;
c23 – 7 <= t23;
c33 – 7 <= t33;
c43 – 7 <= t43;
c14 – 10 <= t14;
c24 – 10 <= t24;
c34 – 10 <= t34;
c44 – 10 <= t44;
c15 – 10 <= t15;
c25 – 10 <= t25;
c35 – 10 <= t35;
c45 – 10 <= t45;
c16 – 18 <= t16;
c26 – 18 <= t26;
c36 – 18 <= t36;
c46 – 18 <= t46;
c17 – 20 <= t17;
c27 – 20 <= t27;
c37 – 20 <= t37;
c47 – 20 <= t47;
c18 – 11 <= t18;
c28 – 11 <= t28;
c38 – 11 <= t38;
c48 – 11 <= t48;
c19 – 11 <= t19;
c29 – 11 <= t29;
c39 – 11 <= t39;
c49 – 11 <= t49;

!Constraints (10);
t11 >= 0;
t21 >= 0;
t31 >= 0;
t41 >= 0;
t12 >= 0;
t22 >= 0;
t32 >= 0;
t42 >= 0;
t13 >= 0;
t23 >= 0;
t33 >= 0;
t43 >= 0;
t14 >= 0;
t24 >= 0;
t34 >= 0;
t44 >= 0;
t15 >= 0;
t25 >= 0;
t35 >= 0;
t45 >= 0;
t16 >= 0;
t26 >= 0;
t36 >= 0;
t46 >= 0;
t17 >= 0;
t27 >= 0;
t37 >= 0;
t47 >= 0;
t18 >= 0;
t28 >= 0;
t38 >= 0;
t48 >= 0;
t19 >= 0;
t29 >= 0;
t39 >= 0;
t49 >= 0;

!Declare binary integer variables
@BIN(x11);
@BIN(x21);
@BIN(x31);
@BIN(x41);
@BIN(x12);
@BIN(x22);
@BIN(x32);
@BIN(x42);
@BIN(x13);
@BIN(x23);
@BIN(x33);
@BIN(x43);
@BIN(x14);
@BIN(x24);
@BIN(x34);
@BIN(x44);
@BIN(x15);
@BIN(x25);
@BIN(x35);
@BIN(x45);
@BIN(x16);
@BIN(x26);
@BIN(x36);
@BIN(x46);
@BIN(x17);
@BIN(x27);
@BIN(x37);
@BIN(x47);
@BIN(x18);
@BIN(x28);
@BIN(x38);
@BIN(x48);
@BIN(x19);
@BIN(x29);
@BIN(x39);
@BIN(x49);
@BIN(y112);
@BIN(y113);
@BIN(y114);
@BIN(y115);
@BIN(y116);
@BIN(y117);
@BIN(y118);
@BIN(y119);
@BIN(y123);
@BIN(y124);
@BIN(y125);
@BIN(y126);
@BIN(y127);
@BIN(y128);
@BIN(y129);
@BIN(y134);
@BIN(y135);
@BIN(y136);
@BIN(y137);
```

```
@BIN(y138);
@BIN(y139);
@BIN(y145);
@BIN(y146);
@BIN(y147);
@BIN(y148);
@BIN(y149);
@BIN(y156);
@BIN(y157);
@BIN(y158);
@BIN(y159);
@BIN(y167);
@BIN(y168);
@BIN(y169);
@BIN(y178);
@BIN(y179);
@BIN(y189);
@BIN(y212);
@BIN(y213);
@BIN(y214);
@BIN(y215);
@BIN(y216);
@BIN(y217);
@BIN(y218);
@BIN(y219);
@BIN(y223);
@BIN(y224);
@BIN(y225);
@BIN(y226);
@BIN(y227);
@BIN(y228);
@BIN(y229);
@BIN(y234);
@BIN(y235);
@BIN(y236);
@BIN(y237);
@BIN(y238);
@BIN(y239);
@BIN(y245);
@BIN(y246);
@BIN(y247);
@BIN(y248);
@BIN(y249);
@BIN(y256);
@BIN(y257);
@BIN(y258);
@BIN(y259);
@BIN(y267);
@BIN(y268);
@BIN(y269);
@BIN(y278);
@BIN(y279);
@BIN(y289);
@BIN(y312);
@BIN(y313);
@BIN(y314);
@BIN(y315);
@BIN(y316);
@BIN(y317);
@BIN(y318);
@BIN(y319);
@BIN(y323);
@BIN(y324);
@BIN(y325);
@BIN(y326);
@BIN(y327);
@BIN(y328);
@BIN(y329);
@BIN(y334);
@BIN(y335);
@BIN(y336);
@BIN(y337);
@BIN(y338);
@BIN(y339);
@BIN(y345);
@BIN(y346);
@BIN(y347);
@BIN(y348);
@BIN(y349);
@BIN(y356);
@BIN(y357);
@BIN(y358);
@BIN(y359);
@BIN(y367);
@BIN(y368);
@BIN(y369);
@BIN(y378);
@BIN(y379);
@BIN(y389);
@BIN(y412);
@BIN(y413);
@BIN(y414);
@BIN(y415);
@BIN(y416);
@BIN(y417);
@BIN(y418);
@BIN(y419);
@BIN(y423);
@BIN(y424);
@BIN(y425);
@BIN(y426);
@BIN(y427);
@BIN(y428);
@BIN(y429);
@BIN(y434);
@BIN(y435);
@BIN(y436);
@BIN(y437);
@BIN(y438);
@BIN(y439);
@BIN(y445);
@BIN(y446);
@BIN(y447);
@BIN(y448);
@BIN(y449);
@BIN(y456);
@BIN(y457);
@BIN(y458);
@BIN(y459);
@BIN(y467);
@BIN(y468);
@BIN(y469);
@BIN(y478);
@BIN(y479);
@BIN(y489);

END
```

APPENDIX C. DATA FOR SMALL PROBLEM INSTANCES

Table C.1 Data pertains to the small problem instances used in Chapter 6

8J 4M 2SP	Machine				Job Weight	Job Release Time	Job Due Date
	M1	M21	M22	M3			
Machine Availability	7	8	6	3			
Job Index	Job Processing Time on Machine						
J1	15	∞	∞	6	3	8	9
J21	21	∞	∞	23	2	6	10
J22	20	∞	∞	23	2	6	10
J3	∞	18	18	14	4	6	9
J4	4	∞	∞	19	4	6	9
J51	21	21	21	24	3	1	18
J52	14	28	28	23	3	1	18
J6	8	21	21	∞	4	5	9

8J 5M 2SP	Machine					Job Weight	Job Release Time	Job Due Date
	M1	M21	M22	M31	M32			
Machine Availability	6	14	8	3	4			
Job Index	Job Processing Time on Machine							
J11	13	28	28	27	27	4	6	26
J12	15	24	24	22	22	4	6	26
J2	∞	28	28	∞	∞	3	8	29
J3	5	11	11	∞	∞	3	7	30
J41	15	21	21	∞	∞	1	3	26
J42	15	23	23	∞	∞	1	3	26
J5	4	28	28	20	20	3	3	18
J6	18	12	12	15	15	1	4	20

9J 3M 2SP	Machine			Job Weight	Job Release Time	Job Due Date
	M1	M2	M3			
Machine Availability	5	7	1			
Job Index	Job Processing Time on Machine					
J1	11	27	∞	2	6	23
J2	9	19	∞	4	11	26
J3	23	14	6	2	3	30
J4	∞	∞	23	4	6	30
J51	25	∞	23	4	6	25
J52	21	∞	23	4	6	25
J61	19	28	∞	1	6	25
J62	25	19	∞	1	6	25
J7	11	∞	∞	4	4	22

	Machine Availability	Machine					Job Weight	Job Release Time	Job Due Date
		M1	M21	M22	M31	M32			
		3	2	6	6	9			
9J 5M 2SP	Job Index	Job Processing Time on Machine							
	J1	17	17	17	11	11	1	4	19
	J2	20	∞	∞	∞	∞	1	4	35
	J3	6	13	13	10	10	3	5	8
	J41	∞	22	22	24	24	4	7	36
	J42	∞	20	20	15	15	4	7	36
	J51	19	27	27	18	18	3	2	34
	J52	18	28	28	17	17	3	2	34
	J6	14	27	27	∞	∞	3	4	38
	J7	∞	∞	∞	14	14	3	8	26

	Machine Availability	Machine			Job Weight	Job Release Time	Job Due Date
		M1	M2	M3			
		7	3	2			
10J 3M 2SP	Job Index	Job Processing Time on Machine					
	J1	11	11	∞	3	0	15
	J2	16	∞	25	4	4	13
	J3	9	12	∞	4	4	15
	J4	24	10	∞	4	5	16
	J51	∞	22	27	1	4	15
	J52	∞	22	24	1	4	15
	J61	18	∞	27	3	4	13
	J62	23	∞	24	3	4	13
	J7	8	∞	∞	1	7	14
	J8	7	7	20	1	5	15

	Machine Availability	Machine				Job Weight	Job Release Time	Job Due Date
		M11	M12	M2	M3			
		3	2	6	6			
10J 4M 2SP	Job Index	Job Processing Time on Machine						
	J1	∞	∞	∞	18	1	4	18
	J2	∞	∞	13	27	1	4	48
	J3	13	13	14	13	3	5	46
	J4	∞	∞	10	26	4	7	50
	J51	∞	∞	16	23	3	2	37
	J52	∞	∞	19	21	3	2	37
	J61	∞	∞	22	27	3	4	23
	J62	∞	∞	23	23	3	4	23
	J7	∞	∞	∞	26	3	8	43
	J8	21	21	23	23	4	3	40

	Machine Availability	Machine				Job Weight	Job Release Time	Job Due Date
		M11	M12	M2	M3			
		3	6	2	3			
11J 4M 2SP	Job Index	Job Processing Time on Machine						
	J11	28	28	∞	19	3	3	47
	J12	23	23	∞	17	3	3	47
	J2	18	18	23	9	1	6	40
	J3	∞	∞	∞	15	1	9	36
	J4	∞	∞	∞	22	2	6	43
	J5	12	12	13	19	4	1	42
	J6	21	21	∞	12	3	6	24
	J7	21	21	13	14	2	2	54
	J8	27	27	15	20	2	5	52
	J91	25	25	18	15	4	2	36
	J92	24	24	16	17	4	2	36

	Machine Availability	Machine M1	M2	M3	Job Weight	Job Release Time	Job Due Date
		5	5	6			
	Job Index	Job Processing Time on Machine					
12J 3M 2SP	J1	21	14	9	4	3	45
	J21	13	19	21	3	7	26
	J22	14	16	19	3	7	26
	J3	22	∞	17	2	6	59
	J4	9	12	∞	2	5	48
	J5	10	∞	∞	3	2	49
	J6	9	9	∞	2	7	24
	J71	19	23	∞	2	4	49
	J72	19	18	∞	2	4	49
	J8	9	6	∞	1	6	57
	J9	13	∞	∞	3	6	33
	J10	8	∞	∞	2	3	50

	Machine Availability	Machine M11	M12	M2	M3	Job Weight	Job Release Time	Job Due Date
		4	8	6	3			
	Job Index	Job Processing Time on Machine						
12J 4M 2SP	J11	∞	∞	23	22	1	1	22
	J12	∞	∞	23	18	1	1	22
	J2	25	25	∞	∞	4	4	16
	J3	18	18	14	21	1	9	14
	J41	∞	∞	15	19	2	3	18
	J42	∞	∞	18	27	2	3	18
	J5	∞	∞	21	∞	4	2	15
	J6	∞	∞	18	∞	1	2	14
	J7	27	27	∞	10	2	3	16
	J8	∞	∞	20	27	1	7	15
	J9	24	24	18	16	2	4	16
	J10	∞	∞	21	18	2	7	16

	Machine Availability	Machine M11	M12	M13	M2	M31	M32	Job Weight	Job Release Time	Job Due Date
		8	5	5	2	6	5			
	Job Index	Job Processing Time on Machine								
15J 6M 2SP	J1	∞	∞	∞	17	18	18	3	3	16
	J2	∞	∞	∞	6	∞	∞	1	2	15
	J3	∞	∞	∞	20	∞	∞	2	9	11
	J4	∞	∞	∞	14	22	22	4	4	17
	J5	∞	∞	∞	9	13	13	2	6	13
	J6	∞	∞	∞	13	17	17	2	7	28
	J7	∞	∞	∞	14	16	16	4	4	14
	J8	15	15	15	11	∞	∞	2	3	34
	J9	10	10	10	10	17	17	1	3	14
	J10	∞	∞	∞	11	14	14	4	3	6
	J111	21	21	21	16	22	22	4	3	6
	J112	19	19	19	15	14	14	4	3	6
	J121	∞	∞	∞	16	20	20	2	5	16
	J122	∞	∞	∞	15	19	19	2	5	16
	J13	∞	∞	∞	16	∞	∞	1	7	36

	Machine Availability	Machine					Job Weight	Job Release Time	Job Due Date
		M11	M12	M13	M2	M3			
		3	2	6	6	9			
	Job Index	Job Processing Time on Machine							
17J 5M 3SP	J1	∞	∞	∞	11	∞	4	3	73
	J21	∞	∞	∞	27	18	1	6	71
	J22	∞	∞	∞	24	26	1	6	71
	J3	∞	∞	∞	∞	9	4	6	72
	J41	23	23	23	20	∞	3	6	73
	J42	28	28	28	28	∞	3	6	73
	J5	12	12	12	11	10	4	8	72
	J6	11	11	11	14	14	1	4	73
	J7	11	11	11	18	9	2	5	70
	J8	∞	∞	∞	∞	16	3	7	71
	J9	∞	∞	∞	12	16	2	7	57
	J10	14	14	14	21	25	2	4	73
	J11	28	28	28	22	11	3	5	72
	J121	∞	∞	∞	27	26	2	6	72
	J122	∞	∞	∞	21	19	2	6	72
	J13	24	24	24	16	∞	1	5	72
	J14	18	18	18	9	17	4	1	69

Note: Each of these problem instances uses $q_{j1j2} = 1$.

APPENDIX D. EXPERIMENTAL DATA

Appendix D.1 Experimental Data Generated for All Problem Structures

PT = processing times of jobs; Wgt = weight of jobs; RT = release time of jobs; DD = due date of jobs

Table D.1 Data for small problem structure

9 Jobs and 4 Machines Block 1

Machine Type			M1	M2		M3
Units			1	1		2
Availability			6	2		7, 5
PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
20	9	∞	J1	3	9	45
21	16	∞	J21	2	9	40
14	22	∞	J22	2	9	40
15	4	18	J3	2	5	25
18	5	12	J4	3	6	41
5	22	20	J5	2	5	11
16	∞	∞	J6	4	3	31
13	∞	∞	J7	3	4	40
21	4	∞	J8	3	5	43

9 Jobs and 4 Machines Block 3

Machine Type			M1	M2		M3
Units			2	1		1
Availability			8, 1	5		6
PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
19	18	∞	J11	4	6	10
23	18	∞	J12	4	6	10
∞	∞	20	J2	1	2	13
5	∞	15	J3	1	7	10
∞	∞	22	J4	1	5	9
20	∞	∞	J5	4	3	10
∞	6	4	J6	1	2	9
∞	∞	6	J7	3	6	10
9	13	14	J8	2	3	9

9 Jobs and 4 Machines Block 2

Machine Type			M1	M2		M3
Units			1	2		1
Availability			5	5, 3		4
PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
∞	13	6	J1	1	7	35
∞	∞	20	J2	1	4	31
12	14	∞	J3	2	5	17
23	∞	12	J4	3	6	29
∞	12	14	J5	3	1	29
24	∞	21	J61	1	10	22
23	∞	21	J62	1	10	22
12	27	5	J7	1	4	26
9	30	9	J8	3	5	22

9 Jobs and 4 Machines Block 4

Machine Type			M1	M2		M3
Units			1	2		1
Availability			5	3, 1		6
PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
9	∞	25	J1	4	8	47
20	∞	15	J2	3	6	48
16	23	24	J31	3	5	46
21	23	26	J32	3	5	46
6	16	23	J4	1	10	47
8	21	11	J5	3	4	46
17	∞	24	J6	3	6	46
18	19	20	J7	3	2	46
∞	21	25	J8	3	5	47

9 Jobs and 4 Machines Block 5

Machine Type			M1		M2	M3
Units			1		2	1
Availability			5		5, 5	6
PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
8	17	∞	J1	2	6	8
7	∞	21	J2	3	6	13
4	∞	8	J3	4	4	7
2	∞	23	J4	2	4	25
9	20	12	J5	2	6	9
17	12	17	J6	1	4	9
2	15	∞	J7	3	7	9
17	∞	15	J81	2	3	4
14	∞	19	J82	2	3	4

12 Jobs and 3 Machines Block 1

Machine Type			M1		M2	M3
Units			1		1	1
Availability			6		5	7
PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
15	21	∞	J1	1	7	83
8	4	∞	J2	3	6	81
∞	∞	27	J3	4	6	75
∞	9	∞	J4	2	8	75
7	2	16	J5	3	2	82
16	3	25	J6	1	6	85
25	∞	20	J71	3	5	77
21	∞	26	J72	3	5	77
11	13	∞	J8	3	1	83
12	4	25	J9	2	7	84
∞	17	20	J101	2	6	77
∞	18	20	J102	2	6	77

12 Jobs and 3 Machines Block 3

Machine Type			M1		M2	M3
Units			1		1	1
Availability			3		3	5
PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
∞	14	8	J1	1	6	21
∞	9	∞	J2	1	8	15
21	7	23	J3	3	4	14
13	6	14	J4	2	3	15
11	3	∞	J5	2	1	17
23	21	∞	J6	4	7	15
22	13	26	J71	2	6	16
16	18	25	J72	2	6	16
∞	20	17	J81	4	8	16
∞	13	18	J82	4	8	16
17	18	∞	J9	1	4	16
∞	15	∞	J10	4	4	16

12 Jobs and 3 Machines Block 2

Machine Type			M1		M2	M3
Units			1		1	1
Availability			5		5	6
PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
21	14	9	J1	4	3	45
13	19	21	J21	3	7	26
14	16	19	J22	3	7	26
22	∞	17	J3	2	6	59
9	12	∞	J4	2	5	48
10	∞	∞	J5	3	2	49
9	9	∞	J6	2	7	24
19	23	∞	J71	2	4	49
19	18	∞	J72	2	4	49
9	6	∞	J8	1	6	57
13	∞	∞	J9	3	6	33
8	∞	∞	J10	2	3	50

12 Jobs and 3 Machines Block 4

Machine Type			M1		M2	M3
Units			1		1	1
Availability			3		1	11
PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
23	17	∞	J1	4	2	63
∞	5	∞	J2	2	2	78
15	11	22	J3	2	7	79
11	14	19	J4	4	7	78
23	18	∞	J51	4	7	81
16	23	∞	J52	4	7	81
9	12	11	J6	4	7	78
23	23	∞	J71	2	6	80
21	15	∞	J72	2	6	80
20	20	∞	J8	3	7	80
21	24	∞	J9	2	3	79
∞	24	∞	J10	2	8	81

12 Jobs and 3 Machines Block 5

Machine Type	M1	M2	M3
Units	1	1	1
Availability	5	3	1

PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
19	∞	∞	J1	2	2	6
∞	16	6	J2	3	4	6
17	25	17	J31	1	4	9
24	29	17	J32	1	4	9
22	24	22	J41	3	5	17
22	28	21	J42	3	5	17
∞	∞	15	J5	4	5	4
9	∞	7	J6	4	2	16
11	∞	∞	J7	2	2	68
∞	∞	8	J8	4	5	6
23	14	24	J9	3	3	15
24	∞	∞	J10	2	2	50

17 Jobs and 5 Machines Block 1

Machine Type	M1	M2	M3
Units	1	1	3
Availability	5	5	6, 4, 3

PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
7	25	∞	J1	1	7	40
∞	20	19	J21	4	6	74
∞	17	22	J22	4	6	74
17	6	∞	J3	1	8	25
∞	24	∞	J4	1	12	33
∞	16	17	J5	2	5	68
∞	17	9	J6	3	6	58
∞	20	19	J71	4	6	76
∞	19	20	J72	4	6	76
15	17	15	J8	4	3	77
21	∞	∞	J9	4	4	68
20	17	∞	J10	2	8	74
12	11	16	J11	3	4	71
13	20	∞	J12	1	8	76
26	17	∞	J13	1	5	65
∞	8	19	J14	2	3	77
∞	13	∞	J15	4	3	77

17 Jobs and 5 Machines Block 2

Machine Type	M1	M2	M3
Units	1	2	2
Availability	4	5, 7	4, 2

PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
19	∞	22	J11	4	5	26
18	∞	26	J12	4	5	26
10	∞	∞	J2	2	8	23
17	∞	∞	J3	2	5	35
3	∞	26	J4	4	11	55
20	12	∞	J5	4	7	45
17	22	12	J6	4	6	51
7	∞	∞	J7	4	6	27
7	∞	12	J8	4	4	49
20	15	25	J9	3	4	56
∞	19	27	J10	4	5	38
6	∞	∞	J11	3	7	40
17	∞	∞	J12	3	4	50
16	24	∞	J131	3	5	24
16	18	∞	J132	3	5	24
12	22	∞	J14	4	4	32
10	∞	∞	J15	2	5	22

17 Jobs and 5 Machines Block 3

Machine Type	M1	M2	M3
Units	2	2	1
Availability	6, 5	6, 2	8

PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
22	∞	16	J1	4	4	15
∞	25	8	J2	4	6	14
8	17	2	J3	4	8	14
15	19	∞	J4	2	3	14
7	∞	12	J5	3	5	16
10	∞	11	J6	2	6	13
17	∞	18	J71	4	3	14
16	∞	12	J72	4	3	14
19	∞	5	J8	1	3	14
16	∞	21	J91	4	7	14
21	∞	19	J92	4	7	14
10	∞	7	J10	2	8	15
23	∞	15	J11	3	1	15
17	21	10	J12	4	5	14
7	20	10	J13	4	4	14
15	∞	9	J14	3	3	15
17	∞	12	J15	4	4	15

17 Jobs and 5 Machines Block 4

Machine Type	M1	M2	M3
Units	2	2	1
Availability	1, 5	8, 4	5

PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
22	28	15	J1	3	2	70
26	25	8	J2	4	7	55
26	23	22	J3	1	7	69
24	13	10	J4	3	1	70
16	∞	12	J5	3	6	70
24	20	18	J6	1	4	71
∞	22	15	J71	3	4	68
∞	24	18	J72	3	4	68
8	∞	10	J8	3	7	71
∞	∞	12	J9	3	6	69
7	∞	22	J10	3	3	69
20	∞	23	J11	4	4	72
20	27	19	J12	1	2	60
19	21	22	J131	3	8	69
20	23	24	J132	3	8	69
26	20	20	J14	1	6	71
20	∞	11	J15	1	8	71

17 Jobs and 5 Machines Block 5

Machine Type	M1	M2	M3
Units	3	1	1
Availability	4, 5, 5	2	7

PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
∞	16	20	J1	1	7	4
∞	7	12	J2	4	7	6
11	22	7	J3	2	4	14
∞	∞	8	J4	1	3	31
27	25	11	J5	1	1	9
19	11	21	J6	1	6	12
16	23	7	J7	3	2	4
15	18	10	J8	3	8	6
21	23	10	J9	3	5	15
21	17	15	J10	4	5	12
17	∞	24	J11	1	4	4
∞	18	23	J12	2	3	48
∞	22	11	J13	1	7	11
20	18	21	J141	4	4	63
19	22	16	J142	4	4	63
∞	24	21	J151	1	2	34
∞	23	16	J152	1	2	34

Table D.2 Data for medium problem structure

25 Jobs and 10 Machines Block 1

Machine Type			M1		M2	M3
Units			4		2	4
Availability			4, 5, 3, 9		3, 6	10, 3, 6, 6
PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
∞	22	∞	J1	3	4	42
∞	19	14	J21	2	9	47
∞	19	22	J22	2	9	47
∞	19	8	J3	4	7	44
∞	15	9	J4	2	7	44
8	6	8	J5	1	6	43
27	17	16	J6	3	8	46
∞	19	∞	J71	2	2	49
∞	14	∞	J72	2	2	49
8	14	8	J8	1	5	49
26	∞	17	J9	3	1	37
18	4	9	J10	4	6	49
∞	10	∞	J11	1	5	49
13	∞	11	J12	3	5	49
9	19	6	J13	4	7	20
∞	21	8	J14	3	7	42
21	3	6	J15	1	4	44
23	22	20	J161	2	9	46
27	19	21	J162	2	9	46
∞	19	∞	J17	2	3	49
∞	3	∞	J18	1	4	49
21	4	11	J19	4	5	38
∞	3	∞	J20	1	5	48
8	22	7	J21	4	2	13
∞	22	7	J22	2	7	43

25 Jobs and 10 Machines Block 2

Machine Type			M1		M2	M3
Units			3		3	4
Availability			6, 6, 8		7, 3, 5	1, 5, 9, 4
PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
5	13	16	J1	3	5	32
5	26	19	J2	2	4	27
∞	13	∞	J3	2	4	32
∞	11	13	J4	3	5	18
15	26	14	J5	4	5	26
3	∞	17	J6	4	9	23
7	8	24	J7	4	5	41
6	∞	17	J8	1	4	16
13	18	∞	J91	3	3	39
14	26	∞	J92	3	3	39
19	∞	26	J10	4	7	17
12	20	19	J11	3	9	27
18	26	∞	J121	4	8	36
22	20	∞	J122	4	8	36
7	12	∞	J13	4	1	32
7	∞	19	J14	2	3	38
7	18	23	J15	2	4	28
16	17	∞	J16	2	5	38
7	∞	30	J17	1	0	32
4	13	14	J18	1	1	23
3	25	∞	J19	1	5	26
∞	26	30	J201	1	3	35
∞	27	30	J202	1	3	35
∞	21	27	J21	1	5	29
16	21	∞	J22	3	4	18

25 Jobs and 10 Machines Block 3

Machine Type			M1	M2	M3	
Units			5	2	3	
Availability			7, 8, 7, 5, 5	5, 2	5, 7, 5	
PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
19	10	26	J1	2	4	12
∞	15	26	J2	1	7	11
∞	14	∞	J3	4	2	20
20	14	11	J4	2	1	12
∞	14	19	J51	3	8	13
∞	16	25	J52	3	8	13
22	20	26	J61	3	2	11
24	16	25	J62	3	2	11
28	12	20	J7	4	8	19
∞	22	9	J8	4	6	11
∞	15	24	J9	1	7	15
19	21	27	J10	1	12	12
11	19	26	J11	3	4	11
∞	3	∞	J12	4	3	12
∞	17	24	J131	3	5	11
∞	15	20	J132	3	5	11
24	11	∞	J14	2	2	11
20	∞	24	J15	3	3	12
11	10	∞	J16	2	8	11
17	∞	∞	J17	1	5	13
17	15	∞	J18	1	4	11
∞	22	25	J19	3	6	12
19	10	∞	J20	2	6	17
11	19	∞	J21	1	5	12
∞	22	17	J22	4	5	11

25 Jobs and 10 Machines Block 4

Machine Type			M1	M2	M3	
Units			3	4	3	
Availability			5, 5, 5	5, 1, 2, 4	4, 3, 6	
PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
7	∞	∞	J1	1	2	47
21	∞	13	J2	3	4	49
∞	∞	12	J3	1	5	48
20	27	10	J4	4	6	47
∞	∞	22	J5	1	8	48
20	21	14	J6	1	6	47
9	∞	11	J7	4	5	48
16	∞	13	J8	2	9	49
23	17	16	J9	1	7	44
∞	∞	19	J101	4	9	48
∞	∞	18	J102	4	9	48
∞	11	∞	J11	1	5	47
21	∞	∞	J12	2	8	48
18	11	15	J13	4	4	48
∞	∞	25	J14	1	5	47
24	∞	17	J151	2	6	48
15	∞	17	J152	2	6	48
14	∞	11	J16	3	7	45
10	22	∞	J17	3	6	40
20	22	23	J181	2	5	48
22	23	24	J182	2	5	48
15	∞	24	J19	4	6	47
17	∞	10	J20	1	6	47
6	∞	∞	J21	1	5	47
13	∞	∞	J22	2	7	49

25 Jobs and 10 Machines Block 5

Machine Type			M1	M2		M3
Units			1	7		2
Availability			7	9, 3, 3, 7, 4, 3, 7		8, 2
PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
∞	20	24	J1	3	6	6
∞	∞	13	J2	3	1	6
∞	21	∞	J3	4	5	29
8	∞	14	J4	2	2	14
20	23	15	J5	4	2	7
12	∞	∞	J6	1	8	4
21	∞	19	J7	2	4	10
17	23	∞	J8	4	2	10
∞	14	8	J9	2	4	32
14	∞	∞	J10	3	6	9
∞	21	15	J11	2	3	8
6	11	∞	J12	4	6	3
20	21	∞	J13	1	10	8
20	14	13	J14	1	5	11
∞	11	24	J15	4	3	11
10	∞	18	J16	3	9	7
14	∞	22	J171	1	7	9
16	∞	25	J172	1	7	9
19	∞	21	J181	1	2	3
17	∞	26	J182	1	2	3
6	∞	∞	J19	3	5	3
15	23	8	J20	1	5	5
21	23	∞	J211	2	5	3
21	25	∞	J212	2	5	3
13	∞	∞	J22	1	2	14

35 Jobs and 8 Machines Block 1

Machine Type			M1	M2		M3
Units			3	1		4
Availability			5, 3, 5	4		5, 4, 5, 3
PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
19	23	30	J11	2	8	62
17	15	29	J12	2	8	62
∞	19	∞	J2	2	8	81
5	11	25	J3	4	3	63
17	7	21	J4	1	3	44
19	20	23	J51	2	4	22
24	15	30	J52	2	4	22
23	17	∞	J6	4	1	96
22	∞	18	J7	3	5	93
17	23	23	J81	4	8	79
23	17	30	J82	4	8	79
∞	13	∞	J9	3	4	87
17	15	∞	J10	3	3	95
12	∞	∞	J11	4	6	96
9	11	∞	J12	2	7	52
19	5	21	J13	1	4	84
21	12	∞	J14	3	4	43
19	14	∞	J15	2	2	96
∞	15	∞	J16	1	1	81
19	10	14	J17	2	4	84
23	∞	28	J18	1	2	95
∞	15	∞	J19	2	7	68
17	5	∞	J20	2	6	83
15	21	∞	J211	1	8	88
22	21	∞	J212	1	8	88
10	20	∞	J22	3	3	77
17	16	26	J23	1	4	87
∞	15	∞	J24	1	3	87
20	11	26	J25	1	5	87
6	23	27	J26	3	3	88
10	12	29	J27	1	7	90
22	19	11	J28	4	4	83
∞	8	∞	J29	3	3	93
5	9	∞	J30	2	6	93
5	13	∞	J31	2	0	42

35 Jobs and 8 Machines Block 2

Machine Type	M1	M2	M3
Units	4	1	3
Availability	2, 4, 3, 3	3	9, 7, 6

PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
∞	21	17	J11	4	8	50
∞	13	26	J12	4	8	50
18	12	7	J2	2	4	53
∞	3	13	J3	4	2	38
∞	4	9	J4	4	2	74
24	9	∞	J5	4	4	70
∞	6	24	J6	1	7	65
∞	18	21	J71	4	5	35
∞	16	25	J72	4	5	35
23	15	∞	J8	1	4	49
∞	5	15	J9	1	4	72
∞	12	∞	J10	2	4	73
∞	19	23	J11	2	7	66
25	22	21	J12	3	1	69
11	∞	∞	J13	3	2	66
26	18	∞	J141	3	4	69
26	13	∞	J142	3	4	69
13	∞	18	J15	2	11	64
27	7	24	J16	2	2	26
∞	13	∞	J17	4	5	30
∞	4	11	J18	2	7	30
20	9	13	J19	1	7	62
∞	8	17	J20	2	5	32
16	11	∞	J21	1	6	64
∞	∞	14	J22	4	8	76
10	3	∞	J23	1	7	40
∞	19	10	J24	4	2	72
∞	∞	19	J25	4	5	65
∞	22	∞	J26	4	3	54
∞	10	∞	J27	3	8	58
22	11	24	J28	3	2	27
∞	15	23	J291	2	7	72
∞	20	22	J292	2	7	72
21	9	∞	J30	2	6	50
12	22	21	J31	1	7	64

35 Jobs and 8 Machines Block 3

Machine Type	M1	M2	M3
Units	1	4	3
Availability	4	6, 7, 5, 8	5, 4, 3

PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
9	8	20	J1	3	3	17
∞	∞	21	J2	4	5	17
18	16	19	J31	2	3	20
25	24	25	J32	2	3	20
14	25	6	J4	3	7	19
11	∞	22	J5	3	5	17
23	∞	17	J6	2	5	18
22	17	∞	J7	1	2	20
6	∞	∞	J8	4	11	20
13	∞	15	J9	4	5	18
24	∞	∞	J10	2	3	17
∞	18	∞	J11	1	5	17
9	11	∞	J12	3	3	18
23	16	17	J131	3	7	36
22	25	21	J132	3	7	36
8	∞	25	J14	3	3	17
16	17	7	J15	3	7	18
6	15	12	J16	3	10	17
8	21	10	J17	2	3	20
21	18	10	J18	2	6	20
17	18	24	J19	2	4	17
18	6	9	J20	2	4	19
∞	∞	13	J21	4	6	17
16	∞	20	J22	4	6	20
7	21	∞	J23	4	4	20
17	18	21	J241	1	0	19
19	20	22	J242	1	0	19
∞	24	14	J25	4	1	32
23	20	8	J26	2	7	20
21	16	6	J27	4	1	19
7	13	∞	J28	2	3	17
7	18	17	J29	4	7	19
19	∞	25	J301	2	5	20
24	∞	17	J302	2	5	20
24	∞	21	J31	2	4	17

35 Jobs and 8 Machines Block 4

Machine Type			M1		M2	M3
Units			5		1	2
Availability			2, 7, 5, 4, 5		8	9, 2
PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
∞	18	20	J1	2	3	90
∞	21	18	J2	4	5	91
29	7	8	J3	3	3	92
13	15	19	J4	2	8	92
18	15	6	J5	3	1	87
17	22	18	J6	3	4	89
10	18	∞	J7	1	6	90
17	18	15	J8	1	7	92
17	8	16	J9	4	3	90
20	4	24	J10	1	7	90
∞	17	12	J11	3	4	89
∞	21	9	J12	1	3	87
∞	14	∞	J13	4	4	91
∞	∞	23	J14	4	9	93
∞	17	19	J15	4	6	93
∞	19	20	J161	3	9	90
∞	21	20	J162	3	9	90
24	18	18	J17	1	5	90
29	18	10	J18	1	2	92
17	9	17	J19	1	4	91
29	23	6	J20	1	9	84
28	21	22	J21	2	3	93
∞	23	17	J22	2	2	90
25	16	19	J231	4	9	92
24	23	16	J232	4	9	92
27	18	8	J24	3	5	92
∞	16	19	J25	4	6	91
∞	15	25	J261	2	1	91
∞	14	16	J262	2	1	91
10	5	22	J27	2	3	91
∞	21	23	J281	4	7	89
∞	16	19	J282	4	7	89
18	11	24	J29	1	6	92
∞	14	∞	J30	2	7	93
∞	14	11	J31	4	1	93

35 Jobs and 8 Machines Block 5

Machine Type			M1		M2	M3
Units			4		3	1
Availability			4, 3, 6, 1		4, 9, 2	7
PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
∞	∞	3	J1	4	5	26
9	10	13	J2	3	4	18
16	6	15	J3	1	4	78
∞	∞	13	J4	2	7	5
∞	21	3	J5	4	7	19
∞	16	16	J6	4	4	11
8	∞	2	J7	4	5	14
19	19	12	J8	2	9	12
21	17	14	J91	3	4	8
24	24	14	J92	3	4	8
24	∞	11	J10	1	5	17
18	∞	8	J11	3	5	18
25	14	14	J12	4	5	10
21	17	17	J13	4	1	19
19	18	14	J141	1	7	6
23	22	21	J142	1	7	6
11	21	6	J15	2	5	58
∞	11	18	J16	1	9	9
19	21	6	J17	2	1	9
10	24	∞	J18	2	3	16
23	24	2	J19	3	5	18
17	∞	3	J20	2	3	12
∞	24	18	J211	1	6	19
∞	24	19	J212	1	6	19
18	22	20	J221	1	4	18
20	23	17	J222	1	4	18
∞	7	∞	J23	4	7	14
∞	15	3	J24	1	2	15
18	∞	20	J25	1	5	12
∞	21	∞	J26	2	5	16
∞	∞	12	J27	4	3	12
∞	22	11	J28	1	2	19
16	11	16	J29	1	3	8
∞	11	20	J30	1	6	18
∞	∞	2	J31	1	9	27

45 Jobs and 6 Machines Block 1

Machine Type			M1		M2	M3
Units			2		2	2
Availability			4, 5		4, 1	3, 5
PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
12	20	∞	J1	1	6	118
4	∞	∞	J2	2	2	151
23	∞	9	J3	1	5	151
15	∞	∞	J4	3	6	93
22	∞	16	J5	4	2	159
11	23	∞	J6	1	6	168
23	25	18	J71	2	4	146
23	27	27	J72	2	4	146
16	∞	∞	J8	2	2	122
17	∞	∞	J9	1	3	163
16	24	∞	J101	2	7	143
15	26	∞	J102	2	7	143
19	∞	22	J11	3	5	149
17	∞	∞	J12	1	7	168
4	26	15	J13	3	4	60
9	∞	∞	J14	4	10	163
7	15	∞	J15	3	3	164
22	28	17	J16	2	5	151
∞	∞	24	J17	1	4	110
19	17	17	J18	3	2	167
14	∞	21	J19	1	1	160
12	18	24	J20	4	9	172
16	29	21	J211	1	6	145
17	26	25	J212	1	6	145
21	14	27	J22	2	6	101
11	14	13	J23	4	6	151
14	26	∞	J24	1	1	169
10	27	∞	J25	4	9	163
22	∞	18	J26	3	5	150
15	27	23	J271	2	4	168
15	26	19	J272	2	4	168
5	27	23	J28	3	8	163
12	21	8	J29	4	6	151
∞	28	15	J30	1	3	156
23	∞	12	J31	2	12	110
6	20	27	J32	3	6	158
∞	∞	24	J331	4	7	169
∞	∞	19	J332	4	7	169
23	∞	26	J34	1	6	146
6	∞	26	J35	3	6	160
23	∞	19	J361	2	3	148
22	∞	20	J362	2	3	148
10	29	13	J37	3	3	156
21	28	14	J38	4	8	155
17	∞	8	J39	4	2	162

45 Jobs and 6 Machines Block 2

Machine Type			M1		M2	M3
Units			2		3	1
Availability			6, 7		2, 7, 5	9
PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
12	∞	16	J1	1	0	103
∞	21	∞	J2	1	5	65
∞	∞	25	J3	1	3	61
19	20	12	J4	2	3	52
19	∞	17	J51	4	5	77
20	∞	19	J52	4	5	77
11	∞	∞	J6	4	5	105
21	22	12	J7	1	7	116
23	∞	20	J81	4	6	115
21	∞	25	J82	4	6	115
24	23	18	J9	3	9	78
∞	28	24	J10	1	4	92
∞	∞	7	J11	3	5	80
19	26	24	J12	4	3	131
∞	∞	13	J13	2	8	96
23	25	16	J14	4	8	68
21	22	16	J15	3	7	119
∞	16	9	J16	2	7	106
∞	12	10	J17	2	4	69
21	25	21	J181	3	3	76
24	26	21	J182	3	3	76
24	27	19	J19	1	5	63
20	∞	∞	J20	3	8	146
12	10	8	J21	3	3	110
∞	23	23	J22	3	2	131
8	∞	∞	J23	2	6	134
26	21	17	J241	4	9	106
23	27	24	J242	4	9	106
16	∞	∞	J25	1	2	56
20	∞	20	J26	1	7	60
27	22	15	J27	2	5	69
12	23	13	J28	2	4	78
∞	∞	17	J29	4	8	95
22	∞	18	J30	1	1	71
12	20	∞	J31	2	4	53
∞	∞	25	J32	2	3	102
21	∞	25	J331	2	9	87
20	∞	17	J332	2	9	87
9	12	19	J34	3	3	123
∞	∞	11	J35	2	6	61
13	11	18	J36	2	4	100
∞	12	24	J37	2	7	142
18	23	21	J381	2	5	54
20	20	19	J382	2	5	54
25	∞	20	J39	4	3	86

45 Jobs and 6 Machines Block 3

Machine Type		M1		M2		M3
Units		3		1		2
Availability		2, 6, 7		5		4, 10
PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
∞	19	∞	J1	1	9	39
∞	20	∞	J2	4	4	34
21	17	18	J31	3	6	33
27	20	24	J32	3	6	33
19	16	18	J4	4	4	37
14	∞	∞	J5	1	4	39
∞	13	∞	J6	1	2	40
∞	16	∞	J7	1	2	35
∞	26	13	J8	2	8	35
13	25	10	J9	1	8	37
∞	25	∞	J10	1	4	33
∞	23	23	J11	1	7	36
11	25	∞	J12	3	6	39
27	12	23	J13	3	7	65
23	21	∞	J14	1	7	39
∞	9	∞	J15	1	6	40
∞	9	∞	J16	2	11	49
19	8	∞	J17	2	7	35
∞	14	16	J18	3	8	40
∞	19	21	J19	1	2	36
∞	16	∞	J20	1	3	40
∞	25	27	J21	2	5	37
∞	∞	24	J221	2	2	48
∞	∞	19	J222	2	2	48
∞	14	∞	J23	3	7	34
22	23	25	J241	1	5	36
23	23	18	J242	1	5	36
∞	20	26	J251	2	5	34
∞	26	27	J252	2	5	34
19	7	∞	J26	2	2	35
28	11	21	J27	4	0	39
∞	20	26	J28	4	7	34
∞	23	15	J29	4	4	35
∞	18	22	J301	1	4	39
∞	25	23	J302	1	4	39
∞	9	23	J31	1	7	40
24	12	∞	J32	1	7	37
∞	21	26	J33	2	4	36
∞	9	22	J34	4	5	39
∞	∞	25	J35	3	8	34
26	15	18	J36	1	6	39
∞	26	∞	J37	3	12	35
27	18	27	J381	2	3	62
24	25	21	J382	2	3	62
∞	22	9	J39	4	8	34

45 Jobs and 6 Machines Block 4

Machine Type		M1		M2		M3
Units		2		2		2
Availability		7, 6		6, 7		6, 7
PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
24	28	∞	J1	4	9	162
14	18	∞	J2	3	6	165
26	∞	23	J3	4	4	166
16	19	23	J4	4	5	164
19	∞	22	J5	4	9	167
28	19	30	J61	4	1	142
26	24	24	J62	4	1	142
16	12	∞	J7	3	12	162
20	14	∞	J8	3	4	165
21	∞	12	J9	1	5	163
14	23	∞	J10	2	4	160
12	16	24	J11	2	4	165
10	23	17	J12	1	8	151
10	14	∞	J13	4	2	162
26	27	∞	J14	1	3	156
23	∞	28	J151	2	7	160
19	∞	24	J152	2	7	160
25	24	24	J161	2	3	161
21	27	26	J162	2	3	161
21	∞	∞	J17	2	6	148
9	22	∞	J18	2	0	166
21	21	∞	J19	3	3	161
21	25	∞	J201	4	7	130
24	28	∞	J202	4	7	130
22	23	25	J21	2	1	150
26	11	∞	J22	3	2	165
21	11	∞	J23	3	6	150
28	18	29	J24	1	3	166
9	12	13	J25	1	6	164
21	∞	∞	J261	3	5	166
28	∞	∞	J262	3	5	166
11	9	∞	J27	4	4	161
17	∞	11	J28	2	8	163
13	17	∞	J29	4	3	134
15	11	18	J30	4	1	160
14	∞	∞	J31	1	9	139
27	∞	27	J32	4	4	164
28	9	∞	J33	2	4	166
25	25	∞	J34	4	3	131
20	21	∞	J351	4	5	151
24	24	∞	J352	4	5	151
14	∞	∞	J36	1	7	160
15	∞	∞	J37	1	2	162
11	10	∞	J38	3	7	164
17	∞	15	J39	2	5	160

45 Jobs and 6 Machines Block 5

Machine Type	M1	M2	M3
Units	2	2	2
Availability	3, 5	6, 2	6, 5

PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
∞	∞	11	J1	4	5	26
∞	∞	21	J2	4	4	8
9	∞	3	J3	1	8	61
7	∞	21	J4	2	4	26
21	∞	8	J5	2	4	8
20	∞	13	J6	1	9	28
10	∞	17	J7	2	3	7
24	∞	∞	J8	2	2	26
∞	29	8	J9	3	12	8
∞	∞	13	J10	2	7	26
∞	∞	21	J11	3	9	9
11	19	13	J12	4	4	21
24	∞	17	J131	3	3	111
18	∞	18	J132	3	3	111
6	∞	9	J14	3	9	133
19	25	16	J15	1	6	19
20	∞	3	J16	1	7	22
∞	∞	14	J171	2	5	22
∞	∞	19	J172	2	5	22
∞	∞	2	J18	3	5	134
24	∞	15	J19	2	7	16
21	23	16	J20	1	7	23
5	∞	5	J21	4	5	16
19	29	∞	J221	3	1	10
17	23	∞	J222	3	1	10
11	∞	∞	J23	4	4	20
∞	∞	20	J24	2	5	25
20	∞	10	J25	1	3	72
21	∞	9	J26	3	5	9
17	∞	14	J271	3	8	26
21	∞	17	J272	3	8	26
23	∞	15	J281	4	3	17
17	∞	19	J282	4	3	17
17	27	17	J291	3	4	98
23	27	21	J292	3	4	98
17	19	2	J30	2	5	22
∞	∞	5	J31	2	8	18
22	19	4	J32	3	3	31
11	28	18	J33	1	3	32
22	∞	6	J34	2	6	20
24	30	10	J35	4	3	10
11	∞	17	J36	4	7	8
∞	∞	3	J37	4	3	12
16	17	4	J38	1	12	16
∞	23	16	J39	1	5	18

Table D.3 Data for large problem structure

50 Jobs and 11 Machines Block 1

Machine Type			M1				M2				M3		
Units			5				2				4		
Availability			3, 3, 6, 2, 2				2, 5				5, 5, 3, 6		
PT on Machine			Job				PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD	M1	M2	M3	Index	Wgt	RT	DD
∞	∞	14	J1	4	5	113	∞	∞	22	J24	4	3	95
∞	∞	20	J2	1	3	109	26	26	23	J25	3	6	24
26	26	∞	J3	1	4	101	12	12	15	J26	3	7	104
∞	∞	9	J4	3	7	100	∞	∞	25	J27	2	7	109
21	21	13	J5	3	6	46	∞	∞	24	J281	4	5	111
∞	∞	15	J6	3	3	45	∞	∞	20	J282	4	5	111
∞	∞	10	J7	1	5	108	28	28	21	J291	1	5	104
24	24	13	J8	4	5	105	28	28	18	J292	1	5	104
27	27	19	J91	1	7	100	∞	∞	16	J30	1	5	83
30	30	27	J92	1	7	100	∞	∞	18	J31	3	7	107
∞	∞	11	J10	2	2	107	17	17	13	J32	4	6	107
∞	∞	20	J11	1	4	104	∞	∞	25	J33	1	5	106
∞	∞	17	J12	3	5	33	28	28	21	J34	3	3	103
∞	∞	21	J131	3	9	104	25	25	21	J351	1	6	97
∞	∞	19	J132	3	9	104	30	30	23	J352	1	6	97
12	12	24	J14	4	5	101	∞	∞	21	J36	2	5	109
14	14	20	J15	1	3	112	22	22	25	J371	4	5	96
19	19	19	J16	4	6	101	22	22	20	J372	4	5	96
20	20	22	J17	1	5	111	17	17	∞	J38	2	3	95
∞	∞	20	J18	2	2	95	∞	∞	16	J39	1	6	106
11	11	11	J19	2	2	79	∞	∞	27	J40	3	7	113
∞	∞	∞	J20	3	2	102	16	16	13	J41	3	7	99
∞	∞	16	J21	1	1	102	∞	∞	26	J42	3	4	106
∞	∞	12	J22	3	7	107	24	24	21	J43	3	6	100
∞	∞	8	J23	4	4	97	∞	∞	10	J44	2	6	110

50 Jobs and 11 Machines Block 2

Machine Type	M1	M2	M3
Units	5	2	4
Availability	6, 6, 3, 7, 3	4, 2	7, 6, 7, 5

PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
18	∞	22	J1	4	5	70
27	20	10	J2	2	4	41
∞	12	17	J3	4	8	36
∞	4	18	J4	4	2	40
∞	∞	12	J5	4	3	63
∞	23	17	J6	3	4	73
12	5	6	J7	2	3	51
∞	5	∞	J8	4	7	29
30	10	24	J9	3	4	66
28	9	19	J10	3	5	36
28	12	7	J11	3	4	42
22	16	23	J121	1	3	49
29	16	23	J122	1	3	49
∞	15	5	J13	2	2	32
24	17	24	J141	1	6	51
27	23	23	J142	1	6	51
∞	8	16	J15	4	8	65
∞	16	∞	J161	4	5	35
∞	23	∞	J162	4	5	35
∞	10	∞	J17	1	7	75
17	4	23	J18	1	6	69
18	20	15	J19	1	3	73
∞	15	21	J20	1	3	69
∞	22	19	J21	2	3	61
24	∞	∞	J22	2	2	55
∞	9	17	J23	2	7	72
19	17	8	J24	3	6	70
28	21	14	J25	1	6	47
25	18	∞	J261	1	5	79
23	20	∞	J262	1	5	79
∞	16	17	J27	1	2	39
∞	23	5	J28	4	11	73
∞	5	∞	J29	1	5	79
∞	22	21	J30	3	6	44
18	23	8	J31	4	2	70
13	16	9	J32	4	8	46
∞	14	∞	J33	3	4	47
28	20	∞	J34	1	4	58
21	21	21	J351	1	4	49
25	22	18	J352	1	4	49
∞	8	22	J36	1	7	37
27	17	18	J37	3	4	33
19	18	6	J38	2	5	77
30	14	23	J391	3	5	64
28	19	17	J392	3	5	64
12	22	12	J40	1	7	77
∞	22	∞	J41	3	5	52
12	∞	23	J42	3	4	33
∞	11	∞	J43	1	4	46
∞	8	∞	J44	1	3	56

50 Jobs and 11 Machines Block 3

Machine Type	M1	M2	M3
Units	4	2	5
Availability	1, 2, 4, 4	1, 3	6, 5, 7, 4, 3

PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
25	15	23	J11	4	4	22
18	16	28	J12	4	4	22
22	∞	∞	J2	2	4	22
20	20	17	J3	4	6	21
19	8	9	J4	1	3	28
23	14	∞	J5	1	7	22
14	∞	∞	J6	2	4	23
14	15	∞	J7	3	5	28
23	7	14	J8	3	7	21
25	21	∞	J91	3	5	19
20	24	∞	J92	3	5	19
19	11	27	J10	3	5	21
∞	∞	∞	J111	1	6	19
∞	∞	∞	J112	1	6	19
∞	∞	24	J12	4	9	19
19	23	13	J13	2	9	20
20	15	19	J14	1	5	21
∞	9	∞	J15	1	4	20
22	∞	14	J16	3	6	33
26	∞	14	J17	4	2	21
15	18	10	J18	4	5	20
14	5	13	J19	2	10	26
10	23	∞	J20	4	8	19
26	7	∞	J21	4	10	21
19	10	14	J22	4	2	19
20	21	∞	J231	1	4	26
24	24	∞	J232	1	4	26
∞	21	23	J24	1	11	19
18	22	∞	J251	2	10	21
21	16	∞	J252	2	10	21
10	∞	14	J26	2	5	28
17	19	12	J27	3	2	20
14	10	18	J28	3	1	22
21	18	∞	J29	4	3	19
10	6	20	J30	3	4	21
∞	∞	26	J31	1	7	19
25	16	∞	J32	2	8	23
∞	6	13	J33	1	4	20
22	22	26	J341	4	2	40
22	23	19	J342	4	2	40
15	21	28	J35	4	4	39
13	∞	14	J36	4	7	23
24	10	∞	J37	3	6	20
12	∞	∞	J38	2	2	22
13	13	∞	J39	1	7	21
7	16	24	J40	4	3	27
23	15	∞	J41	1	5	21
∞	19	∞	J42	3	7	23
21	13	27	J43	4	6	32
21	11	23	J44	3	8	20

50 Jobs and 11 Machines Block 4

Machine Type			M1		M2	M3
Units			3		5	3
Availability			2, 1, 6		4, 7, 8, 1, 6	11, 10, 5
PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
5	∞	16	J1	1	6	85
15	∞	∞	J2	4	3	85
17	12	3	J3	1	5	84
∞	∞	10	J4	4	5	76
∞	∞	16	J51	2	10	82
∞	∞	20	J52	2	10	82
7	∞	13	J6	4	3	70
20	21	∞	J7	1	6	84
∞	∞	5	J8	2	3	85
∞	∞	6	J9	2	8	82
∞	14	∞	J10	4	5	82
9	8	∞	J11	4	2	82
15	21	14	J121	3	6	75
20	26	20	J122	3	6	75
6	12	12	J13	4	5	84
∞	∞	19	J14	3	6	75
∞	∞	5	J15	2	5	84
15	∞	5	J16	3	7	85
∞	21	4	J17	1	6	82
18	14	15	J18	2	4	82
16	12	7	J19	1	6	84
∞	∞	21	J20	2	6	82
18	12	19	J21	4	5	86
14	∞	∞	J22	2	1	82
19	19	17	J23	4	5	83
20	∞	21	J24	3	3	83
13	17	3	J25	1	7	84
∞	∞	15	J261	3	11	84
∞	∞	19	J262	3	11	84
∞	20	5	J27	1	4	84
6	∞	21	J28	1	3	83
5	∞	4	J29	2	7	84
∞	∞	12	J30	4	1	81
∞	26	5	J31	4	5	84
17	26	14	J32	4	5	86
∞	∞	21	J33	1	2	85
∞	∞	20	J341	3	2	84
∞	∞	14	J342	3	2	84
9	∞	11	J35	4	3	78
∞	∞	3	J36	1	6	85
∞	19	7	J37	1	4	85
5	21	20	J38	1	5	84
5	27	18	J39	4	7	84
∞	25	∞	J401	3	1	82
∞	22	∞	J402	3	1	82
∞	8	12	J41	1	7	84
∞	∞	15	J42	3	9	83
∞	∞	17	J43	2	1	75
22	20	17	J441	1	6	82
15	25	20	J442	1	6	82

50 Jobs and 11 Machines Block 5

Machine Type			M1		M2	M3
Units			5		3	3
Availability			5, 5, 4, 2, 8		6, 6, 6	0, 3, 5
PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD
∞	20	17	J11	3	3	5
∞	28	12	J12	3	3	5
28	21	∞	J21	2	3	21
30	19	∞	J22	2	3	21
∞	22	6	J3	3	2	21
26	17	∞	J4	2	4	11
∞	26	∞	J5	2	4	48
28	11	15	J6	3	2	18
24	11	∞	J7	1	3	58
29	19	9	J8	2	5	9
28	17	14	J9	1	7	5
∞	27	19	J10	1	2	11
∞	∞	16	J11	2	6	13
∞	∞	7	J12	2	5	19
17	18	19	J13	1	3	9
19	∞	∞	J14	2	4	10
∞	21	∞	J15	1	6	23
19	∞	14	J16	4	9	18
22	∞	17	J17	3	5	16
12	21	14	J18	1	6	5
25	27	13	J191	3	5	61
30	27	20	J192	3	5	61
24	∞	4	J20	3	2	12
26	24	12	J21	2	5	12
18	15	8	J22	3	4	10
∞	16	20	J23	4	9	13
∞	25	∞	J24	3	7	5
25	21	11	J25	3	8	9
∞	27	∞	J26	3	8	6
29	24	3	J27	3	5	78
26	25	19	J28	4	5	19
25	27	6	J29	1	7	14
16	11	15	J30	1	6	65
∞	23	14	J31	1	5	21
13	27	∞	J32	2	5	5
∞	22	14	J331	3	2	44
∞	26	17	J332	3	2	44
18	10	7	J34	2	4	9
25	19	12	J351	1	5	11
21	26	18	J352	1	5	11
15	15	11	J36	2	5	5
17	23	21	J37	4	7	12
∞	9	5	J38	4	6	18
∞	10	2	J39	3	4	13
∞	16	∞	J40	2	9	21
∞	∞	16	J411	3	7	21
∞	∞	13	J412	3	7	21
27	∞	14	J42	4	5	6
27	23	10	J43	3	4	12
∞	13	11	J44	3	13	19

60 Jobs and 15 Machines Block 1

Machine Type			M1				M2				M3		
Units			5				4				6		
Availability			6, 5, 7, 12, 9				9, 9, 6, 5				1, 3, 3, 6, 2, 7		
PT on Machine			Job				PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD	M1	M2	M3	Index	Wgt	RT	DD
11	4	∞	J1	1	1	78	16	13	12	J26	4	2	54
23	16	∞	J2	3	2	86	6	9	17	J27	1	2	75
7	∞	18	J3	4	3	81	∞	15	24	J28	4	4	57
7	21	18	J4	3	7	35	16	18	22	J29	2	4	78
5	8	∞	J5	1	8	86	21	22	∞	J30	4	8	76
5	14	∞	J6	2	8	88	20	23	26	J31	2	9	83
5	11	17	J7	4	7	84	21	10	∞	J32	4	6	67
∞	11	28	J8	2	3	49	∞	18	20	J33	2	2	80
18	22	26	J91	1	6	82	9	4	∞	J34	1	5	82
24	17	26	J92	1	6	82	24	4	∞	J35	3	4	80
19	16	∞	J101	4	3	84	19	15	∞	J36	4	4	85
18	16	∞	J102	4	3	84	24	19	24	J37	3	5	87
17	19	11	J11	2	4	87	∞	13	13	J38	1	6	79
10	23	15	J12	4	6	48	∞	17	14	J39	4	6	86
20	19	24	J131	2	7	88	20	8	14	J40	3	9	66
17	15	25	J132	2	7	88	∞	22	∞	J41	2	2	87
7	11	∞	J14	1	4	88	20	19	∞	J421	1	4	85
13	5	∞	J15	2	4	76	22	22	∞	J422	1	4	85
18	14	29	J161	3	7	32	21	5	25	J43	2	4	81
18	22	29	J162	3	7	32	5	9	∞	J44	3	5	83
17	21	∞	J171	3	4	77	12	21	∞	J45	4	3	86
17	17	∞	J172	3	4	77	∞	4	∞	J46	3	3	75
16	17	∞	J18	4	5	79	∞	9	21	J47	3	9	81
22	9	29	J19	4	2	84	14	9	16	J48	1	5	46
∞	16	∞	J20	1	5	77	5	14	28	J49	4	5	84
9	11	∞	J21	1	9	80	∞	14	∞	J501	3	2	84
18	18	∞	J22	2	3	19	∞	14	∞	J502	3	2	84
∞	16	∞	J23	3	2	85	∞	14	∞	J511	3	3	88
13	14	∞	J24	4	7	76	∞	18	∞	J512	3	3	88
9	8	∞	J25	2	4	77	16	7	∞	J52	2	5	88

60 Jobs and 15 Machines Block 2

Machine Type			M1				M2			M3			
Units			5				5			5			
Availability			2, 2, 4, 11, 8				6, 2, 3, 1, 9			6, 5, 3, 3, 3			
PT on Machine			Job				PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD	M1	M2	M3	Index	Wgt	RT	DD
18	22	21	J11	3	3	44	∞	23	∞	J262	2	4	28
24	22	21	J12	3	3	44	13	∞	14	J27	1	5	54
∞	∞	9	J2	3	6	59	25	21	∞	J281	2	9	58
20	19	∞	J31	1	6	41	21	22	∞	J282	2	9	58
23	22	∞	J32	1	6	41	12	8	∞	J29	4	6	47
18	9	16	J4	2	3	30	17	20	26	J30	4	2	70
9	7	9	J5	2	7	30	24	7	14	J31	4	6	42
15	14	15	J6	3	6	42	∞	16	∞	J32	3	5	44
14	15	28	J7	1	7	38	∞	17	∞	J331	2	3	63
23	24	∞	J8	3	6	66	∞	16	∞	J332	2	3	63
13	18	12	J9	3	6	42	19	5	27	J34	3	7	48
∞	15	∞	J10	2	7	66	12	18	∞	J35	2	9	68
11	20	24	J11	2	4	51	7	10	9	J36	4	2	43
∞	21	19	J121	2	7	26	15	13	∞	J37	4	8	38
∞	16	27	J122	2	7	26	∞	6	∞	J38	1	5	57
21	∞	∞	J13	4	5	54	25	20	∞	J39	2	6	46
10	13	∞	J14	1	7	68	22	∞	23	J401	1	4	34
∞	16	∞	J15	3	3	53	18	∞	23	J402	1	4	34
14	22	∞	J16	2	2	66	24	10	∞	J41	3	1	51
25	19	16	J17	3	5	33	10	22	20	J42	1	7	67
17	16	15	J18	2	9	34	25	∞	∞	J43	2	3	37
21	11	11	J19	1	2	39	14	8	12	J44	4	7	64
∞	19	∞	J20	2	4	57	7	18	∞	J45	4	4	52
21	23	∞	J211	1	6	42	10	9	∞	J46	3	4	56
22	16	∞	J212	1	6	42	16	7	∞	J47	3	5	31
12	∞	∞	J22	3	4	35	14	17	∞	J48	2	6	58
20	7	14	J23	2	1	27	25	15	∞	J49	2	5	30
23	24	16	J24	1	10	64	∞	18	18	J50	3	4	27
19	23	∞	J25	4	6	53	∞	18	22	J51	2	4	31
∞	20	∞	J261	2	4	28	9	18	25	J52	2	4	44

60 Jobs and 15 Machines Block 3

Machine Type	M1	M2	M3
Units	7	3	5
Availability	4, 5, 10, 5, 11, 4, 4	4, 7, 7	8, 5, 5, 5, 6

PT on Machine			Job				PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD	M1	M2	M3	Index	Wgt	RT	DD
∞	11	∞	J1	2	2	19	∞	16	∞	J27	2	7	17
23	15	15	J21	1	3	18	16	23	∞	J28	4	6	18
23	16	19	J22	1	3	18	25	22	20	J291	4	2	19
15	7	∞	J3	2	2	18	24	22	15	J292	4	2	19
∞	5	15	J4	1	7	26	∞	20	5	J30	3	12	16
27	∞	4	J5	3	7	17	∞	18	∞	J31	3	5	17
17	6	∞	J6	3	4	18	24	15	∞	J32	4	9	18
24	22	∞	J7	1	5	17	14	14	15	J33	3	6	28
∞	10	∞	J8	2	8	16	24	∞	9	J34	3	2	19
∞	23	18	J91	1	5	24	29	15	7	J35	2	7	16
∞	14	18	J92	1	5	24	21	10	23	J36	2	5	31
14	10	10	J10	2	4	17	∞	12	5	J37	1	4	17
∞	∞	22	J11	4	2	16	22	6	∞	J38	2	5	17
10	20	∞	J12	4	7	17	22	7	19	J39	3	6	17
∞	9	∞	J13	4	4	16	∞	10	19	J40	2	6	22
26	4	22	J14	3	5	16	22	14	6	J41	1	7	17
∞	∞	8	J15	1	5	32	25	∞	14	J421	4	6	19
∞	14	∞	J16	2	4	16	23	∞	21	J422	4	6	19
17	22	10	J17	3	7	18	∞	22	∞	J43	2	4	19
22	12	∞	J18	1	3	18	∞	16	14	J441	2	5	17
19	∞	20	J19	2	10	28	∞	17	23	J442	2	5	17
13	12	5	J20	1	8	16	∞	14	9	J45	2	4	19
∞	15	9	J21	2	5	18	25	12	9	J46	4	7	19
20	23	∞	J22	1	3	17	12	10	18	J47	4	5	16
23	11	7	J23	3	2	27	10	11	∞	J48	4	9	16
∞	18	15	J241	4	5	16	∞	10	10	J49	4	6	17
∞	15	17	J242	4	5	16	13	4	5	J50	3	10	16
13	11	21	J25	2	7	18	27	17	17	J511	1	6	16
29	16	19	J261	3	7	17	20	16	18	J512	1	6	16
20	22	16	J262	3	7	17	∞	13	∞	J52	2	4	18

60 Jobs and 15 Machines Block 4

Machine Type			M1				M2			M3			
Units			4				4			7			
Availability			2, 1, 7, 4				6, 3, 9, 4			7, 3, 4, 9, 9, 6, 5			
PT on Machine			Job				PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD	M1	M2	M3	Index	Wgt	RT	DD
21	19	11	J1	3	4	87	∞	18	∞	J261	3	3	87
20	18	26	J21	2	3	88	∞	26	∞	J262	3	3	87
22	20	23	J22	2	3	88	11	18	∞	J27	4	3	88
∞	14	27	J3	4	7	87	10	11	30	J28	2	9	75
∞	13	∞	J4	1	5	89	27	∞	∞	J291	4	6	85
26	13	∞	J5	4	5	88	29	∞	∞	J292	4	6	85
14	∞	∞	J6	4	3	88	∞	20	23	J30	2	11	86
26	15	14	J7	3	5	86	18	24	∞	J31	3	3	88
21	20	∞	J81	4	1	86	25	∞	26	J32	4	5	88
27	21	∞	J82	4	1	86	29	∞	18	J33	2	10	87
10	23	∞	J9	1	6	87	16	19	13	J34	4	7	89
12	11	25	J10	2	5	85	∞	∞	16	J35	2	9	86
11	21	∞	J11	1	4	89	∞	11	∞	J36	4	11	88
∞	24	∞	J121	4	5	84	27	9	13	J37	3	6	89
∞	21	∞	J122	4	5	84	12	26	∞	J38	3	6	88
14	15	21	J13	2	5	87	18	17	∞	J39	2	6	87
12	20	∞	J14	2	4	82	23	22	∞	J40	3	5	87
14	25	24	J15	3	9	86	∞	21	30	J411	2	3	86
25	10	29	J16	2	6	86	∞	23	27	J412	2	3	86
∞	22	∞	J17	3	9	88	∞	∞	17	J42	2	5	86
∞	11	17	J18	1	2	88	14	22	25	J43	4	3	89
27	12	24	J19	1	6	86	21	15	∞	J44	2	6	88
25	19	28	J201	3	6	87	16	13	30	J45	2	6	78
27	18	29	J202	3	6	87	∞	8	11	J46	1	6	86
∞	15	∞	J21	1	6	85	10	∞	30	J47	3	3	88
13	26	23	J22	2	4	87	12	22	21	J48	1	3	77
19	17	∞	J23	3	2	89	17	24	∞	J49	1	3	86
∞	22	26	J241	1	9	86	23	23	∞	J50	1	2	75
∞	20	22	J242	1	9	86	22	25	27	J51	4	5	86
∞	19	∞	J25	2	6	87	29	∞	∞	J52	2	3	90

60 Jobs and 15 Machines Block 5

Machine Type	M1	M2	M3
Units	4	4	7
Availability	7, 5, 6, 3	11, 6, 7, 2	3, 6, 2, 6, 3, 13, 1

PT on Machine			Job				PT on Machine			Job			
M1	M2	M3	Index	Wgt	RT	DD	M1	M2	M3	Index	Wgt	RT	DD
15	∞	∞	J1	1	6	13	∞	16	19	J291	1	9	18
22	7	23	J2	4	6	13	∞	20	22	J292	1	9	18
11	10	15	J3	2	4	15	15	16	24	J30	1	5	7
7	16	17	J4	4	8	77	25	7	∞	J31	4	6	9
∞	20	20	J51	2	2	15	21	12	∞	J32	1	6	18
∞	25	18	J52	2	2	15	∞	21	17	J33	2	3	8
∞	20	12	J6	1	5	16	∞	20	∞	J341	1	9	13
10	23	10	J7	1	5	16	∞	21	∞	J342	1	9	13
12	18	21	J8	4	4	11	∞	11	∞	J35	4	4	5
∞	24	∞	J9	3	5	34	14	18	∞	J36	2	6	16
10	17	∞	J10	3	11	35	26	25	∞	J37	4	8	6
14	21	15	J11	1	8	18	17	12	15	J38	4	5	16
∞	20	21	J12	1	9	14	23	∞	11	J39	2	3	5
9	7	25	J13	2	4	41	∞	20	25	J40	4	5	6
24	6	∞	J14	3	4	7	∞	25	12	J41	1	3	15
20	∞	17	J15	4	3	61	∞	∞	8	J42	1	6	5
21	11	14	J16	4	6	8	16	∞	10	J43	1	5	16
21	20	17	J17	2	9	9	∞	∞	19	J44	1	5	18
∞	15	23	J18	1	4	18	8	12	∞	J45	3	4	7
∞	7	∞	J19	4	5	17	∞	10	23	J46	1	4	14
7	20	∞	J20	3	7	10	24	17	20	J471	1	4	12
21	20	∞	J21	4	6	6	19	25	23	J472	1	4	12
23	10	22	J22	4	8	14	12	12	17	J48	1	6	11
∞	17	15	J23	2	1	15	26	20	∞	J491	3	6	17
17	∞	17	J24	4	3	12	23	25	∞	J492	3	6	17
17	11	∞	J25	3	4	24	22	23	22	J501	1	7	6
∞	22	23	J261	3	5	14	22	19	22	J502	1	7	6
∞	17	25	J262	3	5	14	14	17	∞	J51	3	4	14
13	10	13	J27	3	2	9	18	22	18	J521	2	4	60
14	19	26	J28	4	7	20	21	18	27	J522	2	4	60

Appendix D.2 Tabu Search Parameters

TableD.4 Parameters used in tabu-search based heuristics for each problem structure

Parameters		Small Problem Structures			Medium Problem Structures			Large Problem Structures	
		9J*4M	12J*3M	50J*11M	60J*15M	35J*8M	45J*6M	50J*11M	60J*15M
Tabu List Size	Fixed	6	23	27	15	16	23	23	27
	Variable - Initial - Decrease - Increase	 6 5 7	 5 4 6	 9 7 11	 15 11 18	 16 12 20	 15 11 18	 23 17 27	 27 21 33
Number of Iterations Without Improvement		1	2	3	4	6	7	8	10
Maximum Entries into Index List		2	3	4	6	9	11	13	15
Number of Long Term Restarts		2	2	2	2	2	2	2	2

APPENDIX E. EXPERIMENTAL RESULTS

Table E.1 Experimental results for small problem structure

Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
9 Jobs, 4 Machines, Block 1													
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	163+M	44	5.17	44	16.53	44	18.62	44	9.5	44	28.89	44	27.19
IS2	235	47	6.04	44	17.53	47	15.55	47	11.7	44	34.22	47	23.07
IS3	384	44	6.43	44	18.18	44	21.64	44	11.97	44	33.56	44	31.42
IS4	93	44	3.52	44	7.91	44	12.68	44	6.81	44	19.22	44	14.99
9 Jobs, 4 Machines, Block 2													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	109	74	3.57	74	8.67	74	9.12	74	4.95	74	12.75	74	12.35
IS2	109	74	3.57	74	8.9	74	9.23	74	5.05	74	13.34	74	12.52
IS3	216	84	5.49	84	16.98	84	16.32	84	9.01	84	29.11	84	22.68
IS4	101	74	4.78	74	13.4	74	15.71	74	7.2	74	20.82	74	23.45
9 Jobs, 4 Machines, Block 3													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	418	333	4.67	333	12.08	333	14.62	333	7.63	333	18.62	333	20.05
IS2	418	333	4.72	333	11.86	333	14.34	333	8.07	333	18.51	333	20.87
IS3	1148	333	6.76	333	19.94	333	21.81	333	12.08	333	33.94	333	34.61
IS4	370	333	3.79	333	7.09	333	11.31	333	6.04	333	17.85	333	16.04
9 Jobs, 4 Machines, Block 4													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	77+M	0	4.45	0	12.3	0	14.01	0	6.48	0	18.84	0	20.76
IS2	86	0	4.45	0	12.97	0	15.33	0	6.15	0	19.22	0	20.16
IS3	63	0	3.73	0	11.09	0	12.97	0	5.22	0	16.53	0	17.74
IS4	2	0	3.79	0	7.25	0	11.32	0	5.22	0	14.83	0	16.59
9 Jobs, 4 Machines, Block 5													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	434+M	203	6.7	203	19.67	203	21.81	203	10.49	203	30.26	203	31.53
IS2	450	203	6.64	203	21.26	203	20.55	203	10.55	203	30.65	203	31.58
IS3	560	237	8.57	237	19.66	237	20.43	237	10.93	237	29.93	237	31.14
IS4	255	229	5.88	229	8.95	229	12.63	229	6.1	229	16.7	229	18.18

12 Jobs, 3 Machines, Block 1													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	0+M	0	6.4	0	15.2	0	16.5	0	10.4	0	25.8	0	29.2
IS2	20	0	5.1	0	16.9	0	14.1	0	9.6	0	27.5	0	28.3
IS3	10	0	5.0	0	12.5	0	14.5	0	9.6	0	22.1	0	30.5
IS4	15	0	5.2	0	15.9	0	15.2	0	10.6	0	25.3	0	26.9
12 Jobs, 3 Machines, Block 2													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	455+M	66	11.4	66	39.4	66	35.5	66	19.8	66	72.0	66	56.1
IS2	489	66	11.7	66	37.5	66	35.1	66	21.5	66	62.5	66	57.0
IS3	629	79	9.8	66	31.1	79	31.8	79	20.0	64	55.8	79	46.0
IS4	155	114	6.7	66	18.8	109	19.0	114	13.6	66	34.8	109	33.7
12 Jobs, 3 Machines, Block 3													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	744+M	679	7.1	543	29.9	549	26.4	679	10.3	543	115.2	549	40.3
IS2	753	679	5.2	611	20.7	562	37.0	657	12.3	543	38.7	562	49.5
IS3	951	543	13.4	543	39.8	543	44.9	543	26.0	543	65.4	543	62.3
IS4	725	543	8.7	543	24.7	543	39.2	543	17.5	543	38.1	543	50.2
12 Jobs, 3 Machines, Block 4													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	142+M	40	7.4	26	25.8	23	28.2	40	13.0	26	41.6	23	44.5
IS2	174	40	7.1	40	20.7	23	27.4	40	11.6	40	35.8	23	41.4
IS3	552	12	11.9	12	37.9	12	36.7	12	17.5	12	139.6	12	54.8
IS4	194	48	7.8	28	23.6	40	21.5	48	14.7	28	40.4	40	46.6
12 Jobs, 3 Machines, Block 5													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	926+M	606	10.0	606	28.8	606	40.6	606	21.7	606	59.4	606	50.6
IS2	947	606	9.9	606	28.7	606	35.3	606	27.7	606	87.9	606	54.0
IS3	1334	604	9.2	604	34.7	604	39.3	604	21.1	604	100.6	604	53.3
IS4	762	604	7.2	604	22.2	604	24.9	604	15.3	604	36.5	604	37.5

17 Jobs, 5 Machines, Block 1													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	203	5	13.5	5	48.5	5	65.3	5	17.5	5	58.9	5	76.1
IS2	203	5	13.5	5	47.7	5	67.2	5	17.2	5	59.6	5	79.5
IS3	594	24	13.7	5	48.0	24	51.6	24	17.4	5	58.6	24	67.6
IS4	83	5	15.5	5	49.4	5	49.4	5	20.1	5	64.7	5	62.2
17 Jobs, 5 Machines, Block 2													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	842+M	325	27.6	325	83.3	325	80.1	325	32.9	325	93.4	325	91.9
IS2	850	325	27.1	325	82.2	325	78.6	325	31.6	325	98.9	325	91.8
IS3	2408	353	24.4	349	80.0	353	82.6	353	31.0	349	103.9	353	103.5
IS4	628	325	23.8	325	71.4	325	57.9	325	27.0	325	75.6	325	61.2
17 Jobs, 5 Machines, Block 3													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	1673+ M	1170	42.2	1170	143.5	1160	136.7	1170	50.8	1170	165.1	1160	161.6
IS2	1431	1160	33.5	1156	111.5	1160	104.5	1160	35.9	1156	117.2	1160	117.7
IS3	2075	1195	53.1	1172	144.5	1195	134.8	1195	52.6	1172	158.1	1195	156.5
IS4	1371	1191	24.2	1191	82.7	1191	76.1	1191	28.5	1191	98.3	1191	88.2
17 Jobs, 5 Machines, Block 4													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	54+M	2	25.6	0	85.6	0	75.7	2	30.9	0	102.3	0	93.5
IS2	41	0	19.2	0	60.3	0	56.0	0	22.9	0	77.5	0	66.7
IS3	178	0	23.0	0	84.5	0	76.0	0	27.1	0	96.4	0	92.8
IS4	5	0	19.9	0	62.2	0	49.3	0	23.4	0	74.5	0	59.9
17 Jobs, 5 Machines, Block 5													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	856	438	35.1	438	107.2	438	125.0	438	41.6	438	129.6	438	141.9
IS2	856	438	35.2	438	107.5	438	123.3	438	41.3	438	126.4	438	138.7
IS3	1601	435	42.8	432	141.9	435	156.4	435	53.4	432	158.0	435	189.3
IS4	574	446	29.8	446	100.1	446	72.1	446	34.3	446	102.8	446	85.6

Table E.2 Experimental results for medium problem structure

25 Jobs, 10 Machines, Block 1													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	44	3	74	3	228	3	218	3	96	3	290	3	282
IS2	7	3	53	3	167	2	168	3	74	3	226	2	230
IS3	1392	2	134	2	378	2	393	2	161	2	444	2	464
IS4	46	6	71	5	219	5	235	6	93	5	284	5	295
25 Jobs, 10 Machines, Block 2													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	350+M	140	156	133	538	140	529	140	185	133	598	140	590
IS2	351	138	158	127	627	138	472	138	184	127	620	138	540
IS3	957	150	250	148	795	150	672	150	261	148	873	150	720
IS4	292	129	178	129	620	129	635	129	209	129	740	129	684
25 Jobs, 10 Machines, Block 3													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	1551+M	1094	154	1054	471	1043	416	1094	174	1054	514	1043	494
IS2	1505	1127	143	1085	428	1085	424	1127	166	1093	509	1056	488
IS3	2007	1021	266	1021	573	1021	576	1021	214	1021	643	1021	642
IS4	1119	1020	140	1020	266	1020	309	1020	110	1020	339	1020	381
25 Jobs, 10 Machines, Block 4													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	90+M	20	140	13	402	20	334	20	146	13	446	13	414
IS2	149	18	134	11	360	18	260	18	158	11	429	18	324
IS3	447	7	164	7	479	6	407	7	191	7	561	6	475
IS4	49	23	84	14	340	14	258	23	107	14	392	14	321
25 Jobs, 10 Machines, Block 5													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	1589	1177	145	1177	329	1171	443	1177	188	1177	417	1171	523
IS2	1589	1177	144	1177	331	1171	434	1177	188	1177	416	1171	526
IS3	4593	1180	182	1177	571	1180	543	1180	208	1177	628	1180	640
IS4	1224	1171	74	1171	260	1171	255	1171	96	1171	307	1171	316

35 Jobs, 8 Machines, Block 1													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	273+M	26	267	26	860	22	948	26	705	26	1277	22	908
IS2	125	26	245	26	711	26	756	26	251	26	730	26	771
IS3	4044	22	470	22	1579	22	1433	22	484	22	1566	22	1446
IS4	93	29	329	25	883	29	921	29	336	25	914	29	934
35 Jobs, 8 Machines, Block 2													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	499+M	226	459	223	1219	155	1357	226	391	223	1082	155	1284
IS2	475	117	638	117	1475	117	1677	117	576	117	1441	117	1580
IS3	4689	127	934	127	2227	125	2425	135	795	135	2126	125	2219
IS4	335	150	300	141	1083	150	1013	150	313	141	1123	150	955
35 Jobs, 8 Machines, Block 3													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	874+M	1428	1778	1428	4107	1428	3555	1432	1033	1429	3248	1432	2725
IS2	2747	1510	944	1427	3686	1419	4131	1510	907	1427	3054	1430	2870
IS3	4660	1432	1554	1421	3736	1432	3787	1432	1362	1421	3547	1432	3465
IS4	1622	1419	1095	1414	2919	1419	3194	1419	620	1415	1701	1419	1663
35 Jobs, 8 Machines, Block 4													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	290+M	60	392	60	1164	60	1194	60	408	60	1150	60	401
IS2	390	96	461	55	1225	96	1299	96	480	55	1252	96	1306
IS3	2664	75	578	72	1773	63	1877	75	601	72	1831	63	1893
IS4	215	53	548	53	1458	53	1326	53	569	53	1494	53	1328
35 Jobs, 8 Machines, Block 5													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	2675+M	1123	1014	1123	2666	1123	2703	1123	1200	1123	2839	1123	2472
IS2	2635	1157	801	1129	3134	1148	2591	1157	775	1133	2616	1148	2547
IS3	4064	1144	831	1116	2675	1144	2463	1144	864	1116	2761	1144	2421
IS4	1634	1147	577	1139	1762	1138	1677	1147	543	1139	1670	1138	1636

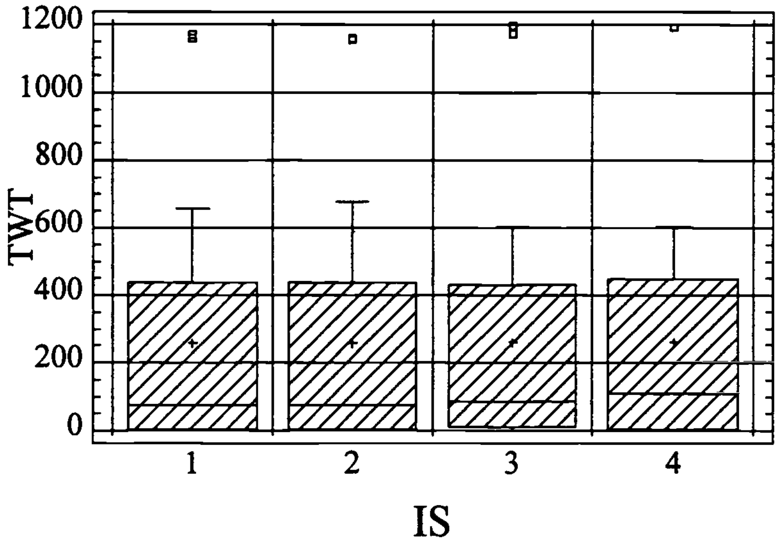
45 Jobs, 6 Machines, Block 1													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	0+M	0	684	0	1343	0	1682	0	803	0	1618	0	2124
IS2	0	0	467	0	1583	0	1396	0	607	0	2772	0	2504
IS3	931	0	944	0	2543	0	2480	0	1010	0	4628	0	4027
IS4	0	0	496	0	1463	0	1580	0	641	0	1890	0	2033
45 Jobs, 6 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+M	410	2216	410	5740	410	5290	410	2205	410	9769	410	7657
IS2	1253	419	2049	405	5617	404	6152	419	2038	405	9184	404	6471
IS3	11578	468	3439	468	8626	409	8379	501	2597	478	8057	409	10643
IS4	658	387	1404	383	4080	382	4960	387	1561	383	6909	382	5094
45 Jobs, 6 Machines, Block 3													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	4784+M	3260	1539	3239	4508	3260	4202	3221	2155	3221	5846	3221	5083
IS2	4885	3259	1453	3259	4219	3259	4398	3259	1586	3259	4611	3259	4858
IS3	18692	3345	2261	3195	7100	3234	6872	3308	2706	3195	7784	3234	7261
IS4	3921	3275	1159	3235	3260	3257	3663	3275	1274	3235	3590	3257	3819
45 Jobs, 6 Machines, Block 4													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	12+M	0	537	0	1562	0	1605	0	662	0	1933	0	1972
IS2	3	0	471	0	1369	0	1427	0	592	0	1730	0	1795
IS3	1367	0	840	0	2525	0	2680	0	963	0	2901	0	3071
IS4	50	0	1013	0	2237	0	2645	0	1149	0	2639	0	2584
45 Jobs, 6 Machines, Block 5													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	4761+M	2725	2532	2712	7596	2660	6964	2725	2687	2712	7952	2660	7714
IS2	4778	2725	2537	2712	6365	2660	6970	2725	2677	2712	6820	2660	7656
IS3	9595	2967	2612	2780	9420	2753	8288	2967	2752	2780	9731	2753	8613
IS4	2856	2763	1177	2763	3794	2763	3601	2763	1128	2763	3655	2763	3907

Table E.3 Experimental results for large problem structure

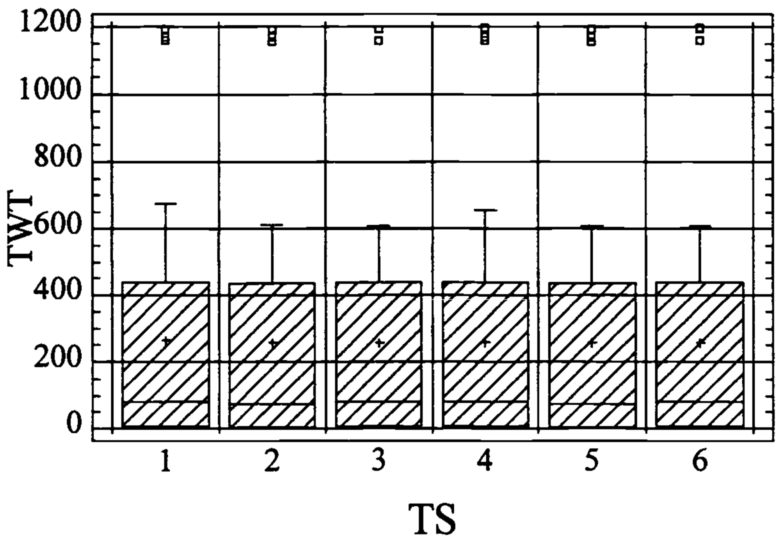
50 Jobs, 11 Machines, Block 1													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	333+M	58	1869	34	6889	53	6811	58	1974	34	6749	53	6575
IS2	484	59	1704	49	6125	46	7855	59	1801	49	6394	46	7430
IS3	6551	50	3202	50	9842	45	9065	50	3306	50	10153	45	9423
IS4	208	58	1449	58	4300	53	4397	58	1533	58	4568	53	4701
50 Jobs, 11 Machines, Block 2													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	702	153	3378	125	9856	76	10507	153	3222	125	9988	76	10179
IS2	751	93	3870	92	12172	93	14417	93	3971	92	12189	93	12519
IS3	4372	130	5131	87	20336	103	13330	84	6655	84	17585	84	14617
IS4	346	189	2926	125	8532	122	10057	189	3049	124	9169	122	10469
50 Jobs, 11 Machines, Block 3													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	3919+ M	2135	3323	1907	14746	1993	11169	1979	4748	1926	12907	1975	15143
IS2	4094	1877	6293	1877	15220	1877	6244	1877	6056	1877	15122	1877	14860
IS3	8677	1931	7771	1931	17985	1931	18404	1932	6901	1932	17099	1932	17807
IS4	2245	1944	4129	1941	7029	1944	7984	1947	1789	1941	4882	1947	5796
50 Jobs, 11 Machines, Block 4													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	106+M	65	1162	31	3723	17	4037	65	1243	31	3968	17	4309
IS2	270	28	2094	15	4826	25	5211	28	2182	15	5086	25	5472
IS3	2244	14	2292	14	6670	14	5906	14	2380	14	6975	14	6203
IS4	141	41	1038	17	4349	25	4305	41	1135	17	4653	25	4527
50 Jobs, 11 Machines, Block 5													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	4557+ M	2549	6957	2546	20739	2484	21063	2560	5220	2547	18130	2490	16590
IS2	4683	2574	11438	2551	24325	2494	27334	2574	6789	2567	17489	2510	19113
IS3	6192	2641	5143	2519	15473	2480	16913	2641	4986	2519	16116	2480	16553
IS4	3042	2493	8972	2473	18205	2493	17877	2507	3077	2474	11534	2507	9681

60 Jobs, 15 Machines, Block 1													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	20+M	8	2557	8	7478	8	7737	8	2989	8	8767	8	9080
IS2	8	8	2164	8	6293	8	6548	8	2596	8	7563	8	7855
IS3	1706	32	4800	32	13905	8	15146	32	5271	32	15059	8	16439
IS4	39	8	2656	8	7773	8	8102	8	3110	8	9454	8	9253
60 Jobs, 15 Machines, Block 2													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	815+M	202	21175	202	41161	155	40376	225	11188	206	35379	155	31148
IS2	771	155	16678	150	45608	155	32693	171	11979	171	30789	171	27998
IS3	2907	140	21987	140	51410	140	45912	144	15316	144	42920	144	40207
IS4	600	186	11076	186	34465	185	36031	180	12524	180	31880	180	32501
60 Jobs, 15 Machines, Block 3													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	4113+M	2194	18728	2181	50755	2194	46789	2207	15020	2181	45017	2195	42765
IS2	4060	2191	21229	2145	47959	2176	51554	2232	15338	2145	42878	2177	39602
IS3	8741	2164	20155	2120	51117	2137	44950	2164	14898	2164	42073	2135	42322
IS4	2671	2128	14381	2128	34945	2128	29707	2136	9564	2136	30197	2136	25826
60 Jobs, 15 Machines, Block 4													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	141+M	0	3538	0	10945	0	11866	0	3911	0	12084	0	12999
IS2	199	0	4042	0	11782	0	15154	0	4434	0	12918	0	14893
IS3	2054	0	5574	0	15883	0	16014	0	5908	0	16892	0	17338
IS4	33	0	2899	0	8281	0	10552	0	3292	0	9393	0	11930
60 Jobs, 15 Machines, Block 5													
Initial Solution		TS1		TS2		TS3		TS4		TS5		TS6	
		TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT	TWT	CT
IS1	4479+M	2441	16372	2417	48415	2404	41958	2441	15355	2430	41650	2403	43783
IS2	4434	2467	16852	2394	47604	2447	43447	2469	15406	2400	45299	2447	41353
IS3	5494	2434	12195	2417	39247	2405	40190	2434	12652	2417	39725	2405	40502
IS4	2519	2392	5425	2376	32784	2385	17061	2392	5883	2380	18812	2385	18659

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

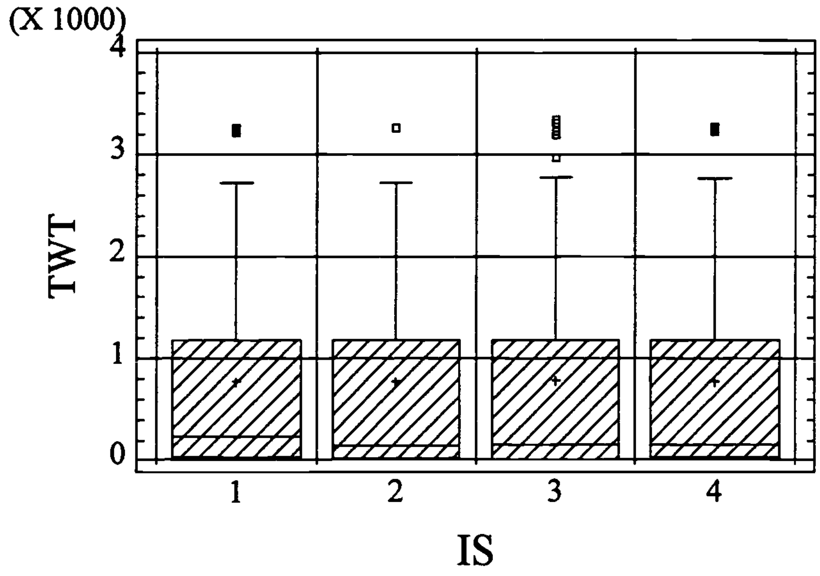


(a)

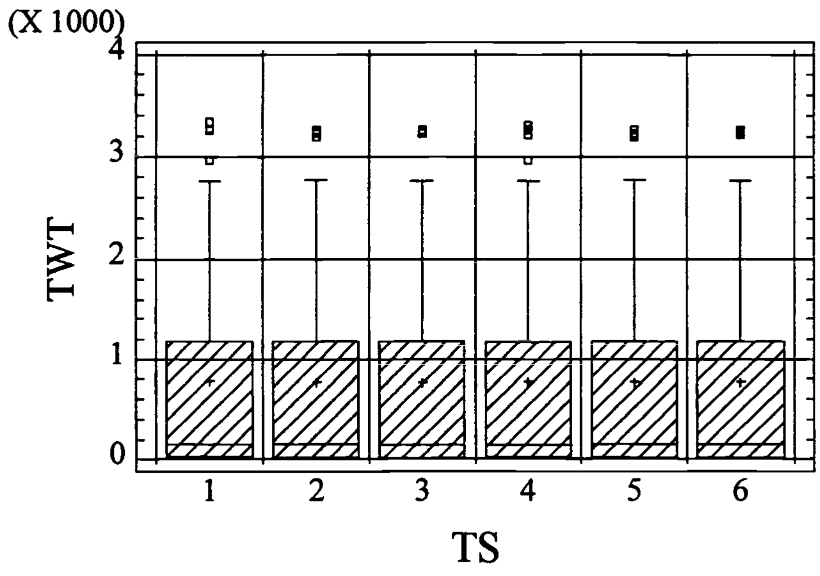


(b)

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

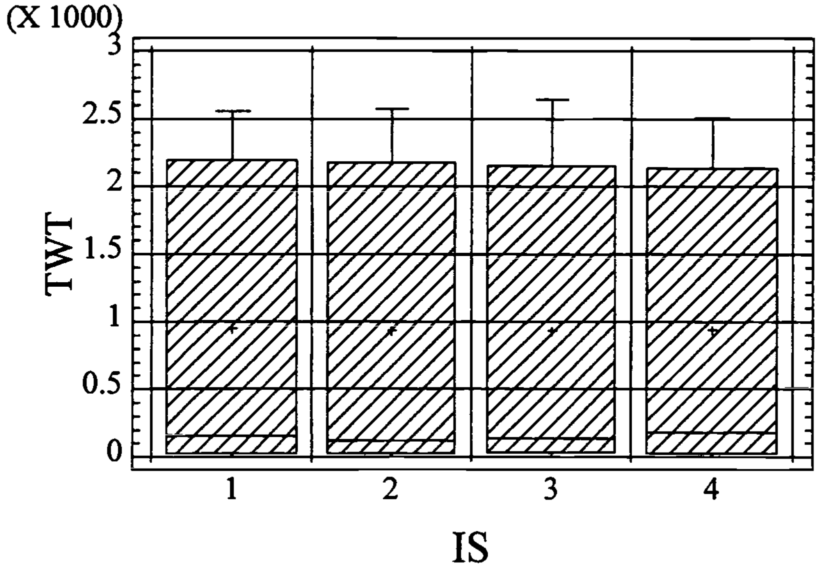


(a)

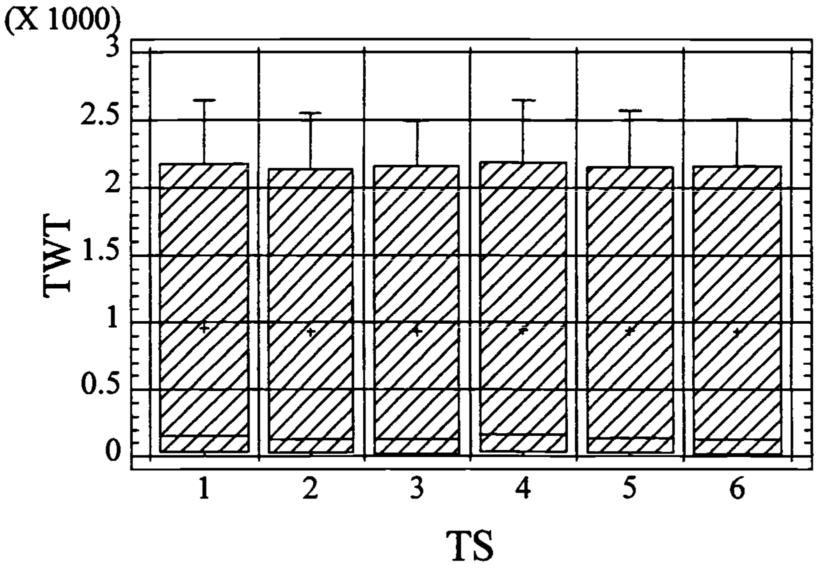


(b)

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



(a)



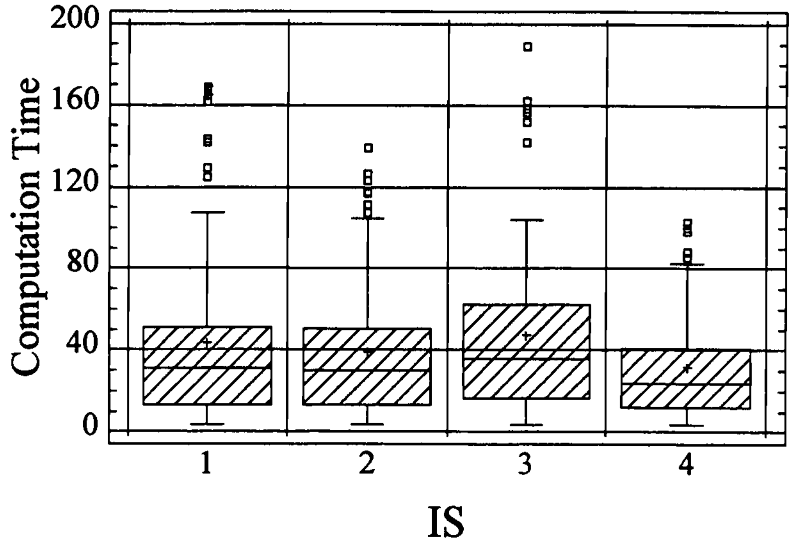
(b)

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

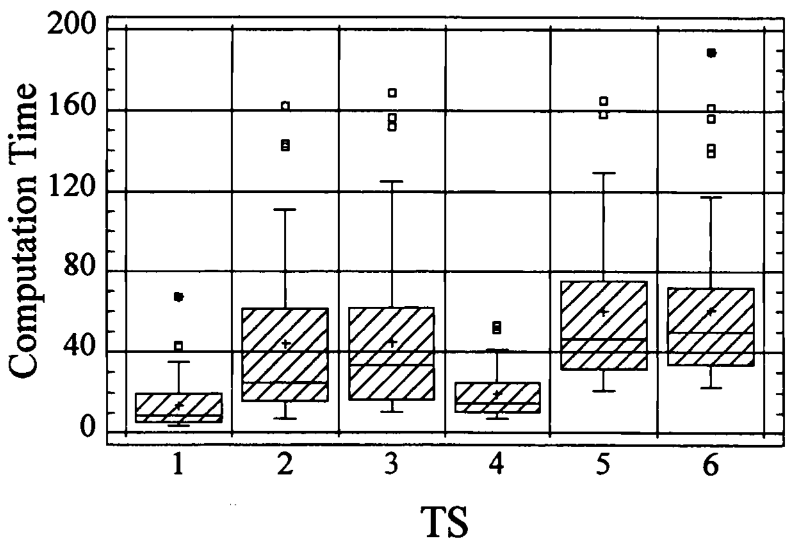
Table F.1 Results of Wilcoxon signed-rank test on total weighted tardiness

Comparisons	Significant Difference at $\alpha = 0.05$		
	Small Problem	Medium Problem	Large Problem
TS1 & TS2	Yes	Yes	Yes
TS1 & TS3	No	Yes	Yes
TS1 & TS4	No	No	No
TS1 & TS5	Yes	Yes	Yes
TS1 & TS6	No	Yes	Yes
TS2 & TS3	No	No	No
TS2 & TS4	Yes	Yes	Yes
TS2 & TS5	No	No	No
TS2 & TS6	No	No	No
TS3 & TS4	No	Yes	Yes
TS3 & TS5	Yes	No	No
TS3 & TS6	No	No	No
TS4 & TS5	Yes	Yes	Yes
TS4 & TS6	No	Yes	Yes
TS5 & TS6	Yes	No	No

APPENDIX G. ANALYSIS OF EXPERIMENTAL RESULTS (COMPUTATION TIME)

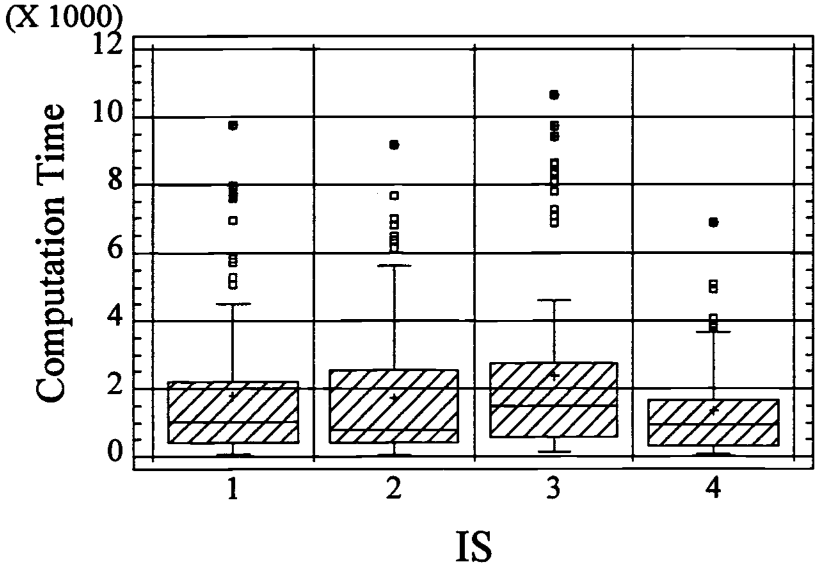


(a)

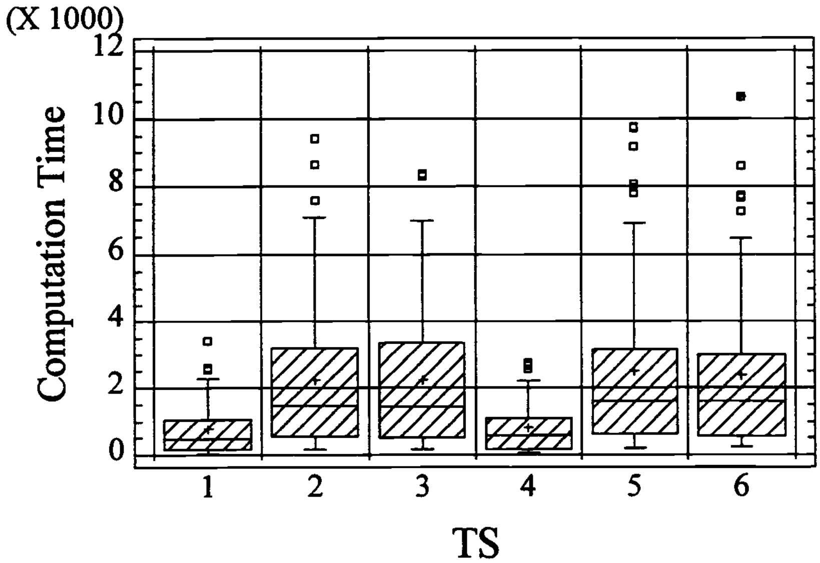


(b)

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

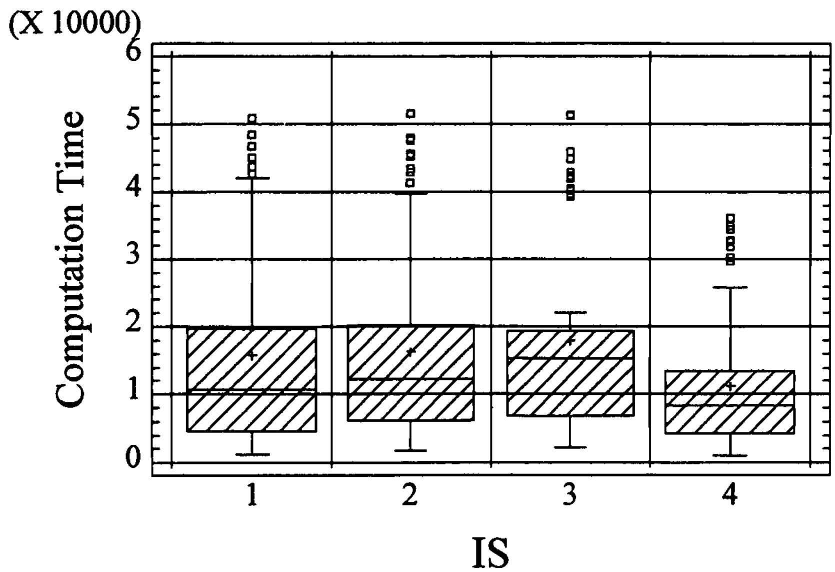


(a)

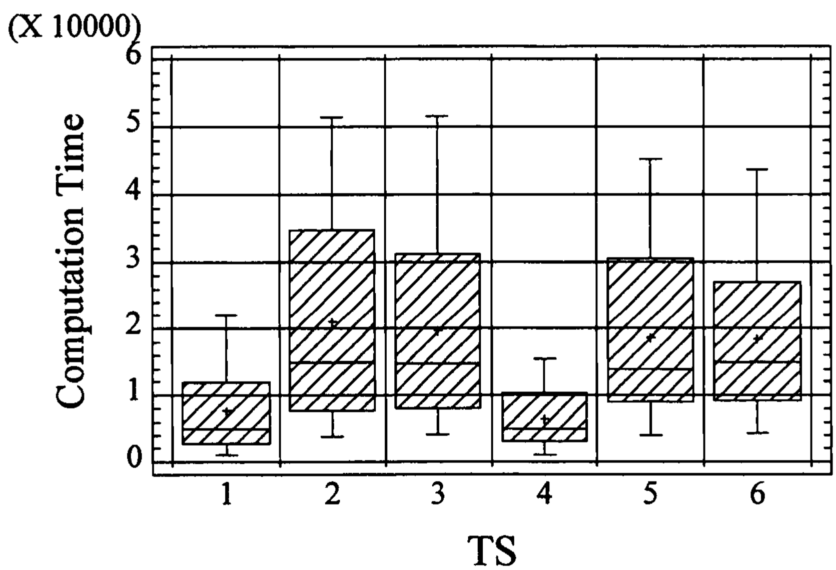


(b)

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

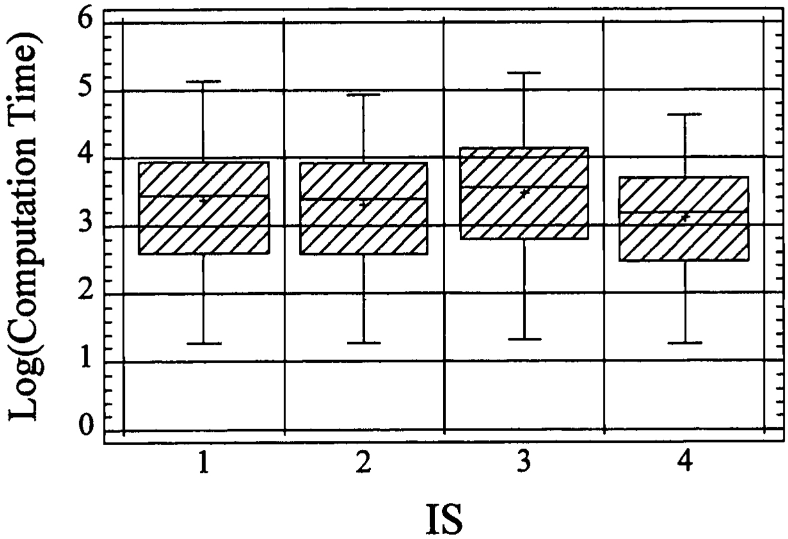


(a)

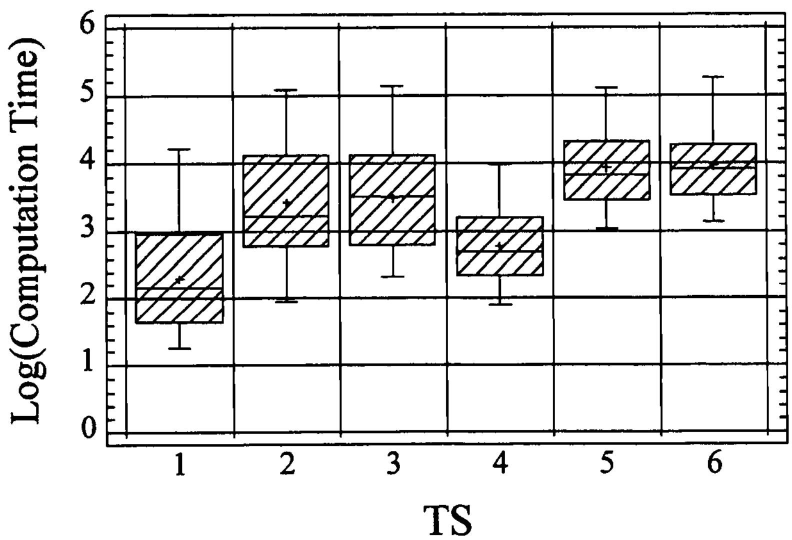


(b)

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

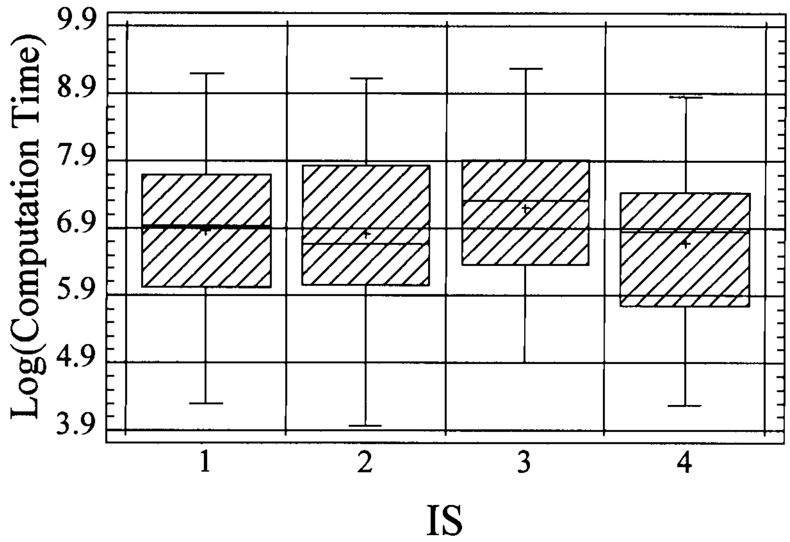


(a)

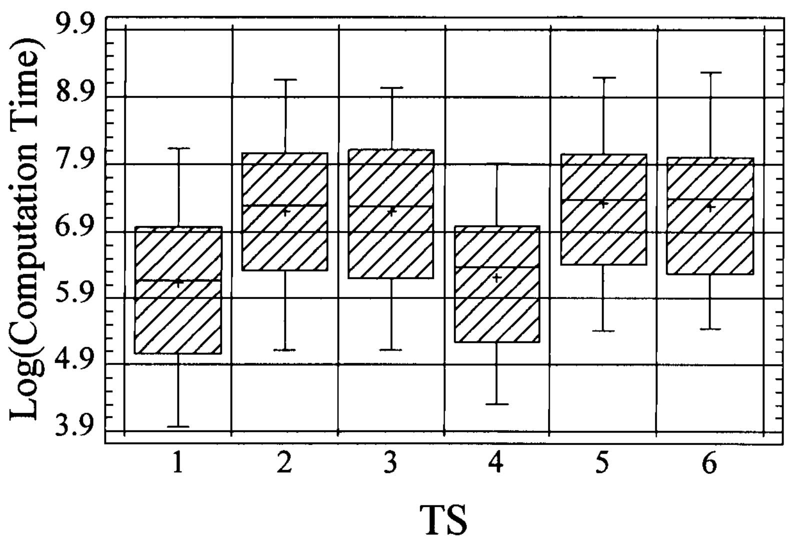


(b)

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

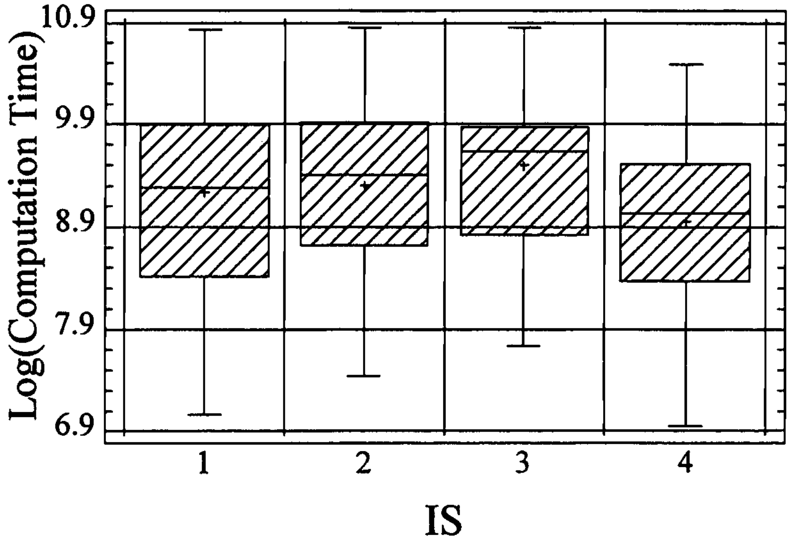


(a)

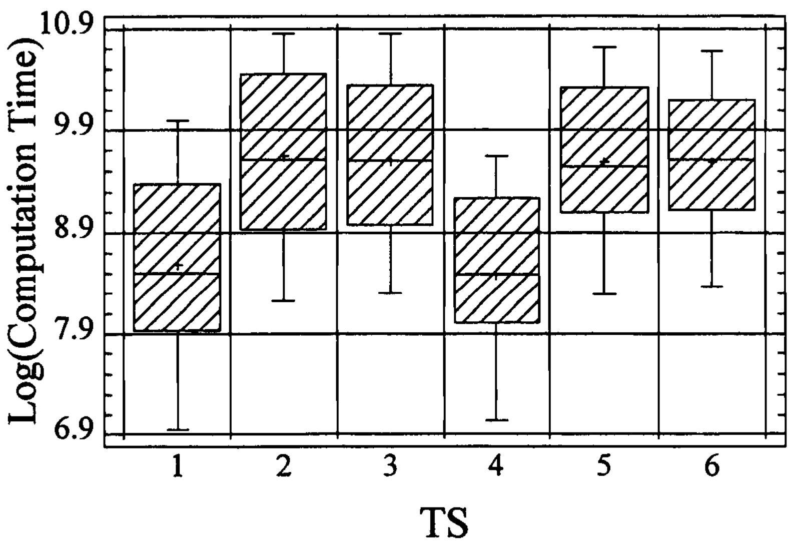


(b)

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



(a)



(b)

Figure G.6 Box Plots of Log(computation time) between (a) levels of IS; (b) levels of TS for large problem structure

Table G.1 ANOVA on Log(computation time) for small problem structure

Source of Variation	Sum of Squares	Degrees of Freedom	Mean Square	F-Ratio	p-value
Whole plot:					
Blocks	145.387	14	10.3848		
IS	6.060	3	2.0198	12.62	< 0.0001
Blocks*IS (whole plot error)	6.720	42	0.1600		
Subplot:					
TS	127.221	5	25.4443	204.70	< 0.0001
Blocks*TS	8.699	70	0.1243		
IS*TS	0.296	15	0.0197	2.10	0.0113
Blocks*IS*TS (subplot error)	1.975	210	0.0094		
Total (corrected)	296.358	359			

Note: F-Ratios are based on the following mean squares:
IS – whole plot error
TS – Blocks*TS
IS*TS – subplot error

Table G.2 ANOVA on Log(computation time) for medium problem structure

Source of Variation	Sum of Squares	Degrees of Freedom	Mean Square	F-Ratio	p-value
Whole plot:					
Blocks	370.001	14	26.4286		
IS	13.089	3	4.3631	21.55	< 0.0001
Blocks*IS (whole plot error)	8.505	42	0.2025		
Subplot:					
TS	91.821	5	18.3641	412.11	< 0.0001
Blocks*TS	3.119	70	0.0446		
IS*TS	0.288	15	0.0192	1.32	0.1912
Blocks*IS*TS (subplot error)	3.051	210	0.0145		
Total (corrected)	489.874	359			

Note: F-Ratios are based on the following mean squares:
IS – whole plot error
TS – Blocks*TS
IS*TS – subplot error

Table G.3 ANOVA on Log(computation time) for large problem structure

Source of Variation	Sum of Squares	Degrees of Freedom	Mean Square	F-Ratio	p-value
Whole plot:					
Blocks	122.785	9	13.6428		
IS	9.367	3	3.1222	14.33	< 0.0001
Blocks*IS (whole plot error)	5.881	27	0.2178		
Subplot:					
TS	60.648	5	12.1295	238.21	< 0.0001
Blocks*TS	2.291	45	0.0509		
IS*TS	0.211	15	0.0141	0.69	0.7867
Blocks*IS*TS (subplot error)	2.738	135	0.0203		
Total (corrected)	203.92	239			

Note: F-Ratios are based on the following mean squares:
IS – whole plot error
TS – Blocks*TS
IS*TS – subplot error

Table G.4 Results of Duncan's analysis on Log(computation time) for small problem structure

TS fixed at	Comparisons	Significant at $\alpha = 0.05$	TS fixed at	Comparisons	Significant at $\alpha = 0.05$
TS1	IS1 & IS2	Yes	TS4	IS1 & IS2	No
	IS1 & IS3	Yes		IS1 & IS3	Yes
	IS1 & IS4	Yes		IS1 & IS4	Yes
	IS2 & IS3	Yes		IS2 & IS3	Yes
	IS2 & IS4	Yes		IS2 & IS4	Yes
	IS3 & IS4	Yes		IS3 & IS4	Yes
TS2	IS1 & IS2	Yes	TS5	IS1 & IS2	Yes
	IS1 & IS3	Yes		IS1 & IS3	Yes
	IS1 & IS4	Yes		IS1 & IS4	Yes
	IS2 & IS3	Yes		IS2 & IS3	Yes
	IS2 & IS4	Yes		IS2 & IS4	Yes
	IS3 & IS4	Yes		IS3 & IS4	Yes
TS3	IS1 & IS2	Yes	TS6	IS1 & IS2	No
	IS1 & IS3	Yes		IS1 & IS3	Yes
	IS1 & IS4	Yes		IS1 & IS4	Yes
	IS2 & IS3	Yes		IS2 & IS3	Yes
	IS2 & IS4	Yes		IS2 & IS4	Yes
	IS3 & IS4	Yes		IS3 & IS4	Yes

Table G.5 Results of Duncan's analysis on Log(computation time) for medium and large problem structures

Comparisons	Significant at $\alpha = 0.05$	
	Medium Problem	Large Problem
IS1 & IS2	Yes	Yes
IS1 & IS3	Yes	Yes
IS1 & IS4	Yes	Yes
IS2 & IS3	Yes	Yes
IS2 & IS4	Yes	Yes
IS3 & IS4	Yes	Yes

APPENDIX H. PSEUDO-CODE FOR TABU-SEARCH BASED ALGORITHM

```
Generate the initial solution
Determine the tabu search parameters
Admit the initial solution to the Candidate List (CL) and the Index List (IL)
Initialize the Long Term Memory matrix (LTM)
Do
{
    Initialize the Tabu List (TL)
    Initialize the Iteration without improvement (IT)
    Evaluate the total weighted tardiness of the initial solution
    Set the Aspiration Level (AL) to the total weighted tardiness of the initial solution
    Set the initial solution as the current seed
    Do
    {
        Generate the neighborhood solutions by applying swap moves and insert moves to the
        current seed
        For each neighborhood solution generated from the current seed
        {
            Evaluate the total weighted tardiness
            If (move ∈ TL and AL is not satisfied)
                Exclude the solution that results from the move
        }
        The best solution ← ∅
        Do
        {
            Identify the neighborhood solution that has the minimum total weighted tardiness
            If (the neighborhood solution ∉ CL)
            {
                The best solution ← the neighborhood solution
                The best move ← the move that results in the neighborhood solution
            }
        } while (the best solution = ∅)
        The next seed ← the best solution
        TL ← the best move
        If (the total weighted tardiness of the best solution < AL), Update AL
        CL ← the best solution
        If (the current seed = local optima)
        {
            IL ← the current seed
            Entries into IL is increased by 1
        }
        If (the next seed < the current seed)
            Iteration without improvement (IT) ← 0
        Else
            Iteration without improvement (IT) is increased by 1
```

Update LTM matrix
The current seed ← the next seed

} while (both IT and entries into IL have not reached the specified numbers)
Identify the new restart by using the LTM matrix
Check the new restart against the previous restarts
Next initial solution ← new restart solution

} while (the number of restart has not reached the specified number)
Terminate the search
Return the solution with the minimum total weighted tardiness in IL as the best solution found so far