Scheduling of Jobs on a Single Machine to Minimize Total Cost

Singu Babu Yerra

A Thesis
in
The Department
of
Mechanical Engineering

Presented in partial fulfillment of the requirements
for the Degree of Master of Applied Science at
Concordia University
Montreal, Quebec, Canada

February 1996

ISBN 0-612-10911-9

Canada

Abstract

Scheduling of Jobs on a Single Machine to Minimize Total Cost

Singu Babu Yerra

This thesis focuses on the problem of minimizing the total of work-in-process, earliness, tardiness, and machine idle costs involved in the sequencing and scheduling of jobs on a single machine. In the most general case that we consider, jobs can have possibly different release times and multiple due dates. This problem is of special interest because of its use in solving general job shop scheduling problems, its application in a Just-In-Time manufacturing environment, and also because of its own intrinsic value.

A special case of that problem is known to be NP-complete in the strong sense and others were able to solve it optimally only for a very small number of jobs. Therefore we concentrate on providing quick and efficient heuristic procedures. We also propose new optimal idle time insertion procedures to insert idle times between successive jobs of a given sequence. We test our heuristics for range of different factors and compare the results to other heuristic procedures, that could be adapted to the problem.

Acknowledgment

The author is extremely thankful to his supervisors Dr. Samir Ammouny and Dr. A. V. Bulgak for their intellectual guidance and moral support throughout this work. It has been indeed a privilege to work under their supervision.

The author would also like to thank the members of the faculty, staff and fellow students of the department of Mechanical Engineering, Concordia University for their time and assistance throughout the course of the work.

I would like to thank Mr. Arun Yellamraju for his help in correcting the text. It is also appropriate to thank my friends Dr. G. B. Reddy, Koganti Rama, Anil and my room mates Sudheer Reddy, Selva for their encouragement & support during this work.

This thesis is dedicated to author's parents, grand mothers, mama, brother, sister, sister in law for their love and encouragement to achieve excellence, without them this would not have been possible.

Table of Contents

CHAPTER 3

Single due date per job and equal release times

CHAPTER 4

Single due date per job and different release times

List of Figures

List of Tables

Nomenclature

n = Number of independent jobs

r_i = Release time of job i

p_i = Processing time of job i

P = Total processing time of n jobs

d = Common due date of all the jobs for single due date problems

d_i = Due date of job i for jobs with single due date

d_{im} = m th due date of job i for jobs with multiple due dates

C_i = Completion time of job i

W_i = Work-in-process of job i

γ = Common work-in-process penalty for all the jobs / unit time

γ_i = Work-in-process penalty of job i / unit time

E_i = Earliness of job i

α = Common earliness penalty for all the jobs / unit time

α_i = Earliness penalty of job i / unit time

T_i = Tardiness of job i

β = Common tardiness penalty for all the jobs / unit time

β_i = Tardiness penalty of job i / unit time

β_{im} = m th tardiness penalty of job i / unit time for jobs with multiple tardiness penalties

M_i = Total machine idle time

μ = Machine idle cost / unit time

T - Tardiness factor

R - Due date range

TC - Total Cost

σ - The schedule obtained from optimal timing algorithm

σ' - The schedule with small difference from σ

σ^* - Optimal schedule

CR - Potential cost reduction

I_j - Idle time before the first job j in schedule σ

I_j' - Idle time before the first job j in schedule σ'

i, j, k, k', u = Job indexes

CHAPTER 1
Introduction

A job scheduling problem is a problem in which we decide the order of all jobs on each machine and determine the starting time of each operation in order to optimize an objective function. Many researchers consider that job scheduling problems are some of the most interesting problems in production analysis. They have received considerable attention from researchers. Most of the problems are very complex and far from being completely solved because of their combinatorial nature (Elsayed 1994).

A broad classification of job scheduling problems will be given below. But first, we shall define several basic terms that we often come across:

Processing Time: A processing time is the length of time which is required for the completion of a job on a particular machine (S. Ashour 1972).

Sequence: A sequence does specify the arrangement of the operations comprising all jobs on all machines. A sequence does not provide the time at which the operations are performed nor the existence of idle times between various operations (S. Ashour 1972).

Schedule: A schedule is a feasible sequence in which the starting and completion times of the operations comprising all jobs on each of the machines, are specified. It also specifies the idle time, if any, between the processing times of each successive operations or jobs (S. Ashour 1972).

Bottleneck: A bottleneck is the department, work station, or operation that restricts the flow of product through the production system. A bottleneck machine restricts the flow of product from upstream machines and starves down stream machines (Mark and Gregory 1991). Bottlenecks can change over time.

Other commonly used terms are introduced in the glossary at the end of this thesis.

1.1 Problem Classification

Numerous schemes have been proposed for categorizing job scheduling problems. The intention of any classification is to provide a semblance of organization so that the major differentiating dimensions of the problem classes are identified. For our purpose, we classify job scheduling problems as follows:

1.1.1 Processing Capacity

Processing capacity is concerned primarily with the number of processing steps associated with each production task or item. A common breakdown for this class is as follows:

- **One Stage, Single Machine Scheduling**

Single machine scheduling problems have received substantial attention because of several reasons. These types of problems are important because of their own intrinsic value, as well as because of their role as building steps for more generalized and complex problems. Sometimes an entire production line may be treated as a single processor for scheduling purposes. In a multi-processor environment, for example, single machine schedules are used to solve job shop scheduling problems by using a shifting bottleneck procedure (Adams 1988, Dauzere-Peres 1993) or a decomposition approach (Raman 1993, Wilhelm 1994), or to organize task assignment to an expensive processor. Furthermore, single machine problems are of fundamental character and allow for some insight and development of ideas when treating more general scheduling problems.

- **One Stage, Parallel Machine Scheduling**

This problem is similar to the one-machine problem in that each task requires a single processing step, but that step may be performed on any of a number of parallel processors. These processors are usually identical, but in some cases their processing capacities may be different.

- **Multi Stage, Flow Shop Scheduling**

For the multistage problem, each task requires processing at a set of distinct processors or machines. In the flow shop problem, there is a strict precedence ordering of the processing steps for a particular task. All tasks are to be processed on the same set of machines with an identical precedence ordering of their processing steps.

- **Multi Stage, Job Shop Scheduling**

Like the flow shop, there is a strict precedence ordering for each task. But unlike the flow shop, jobs can have different precedence orderings and numbers of operations. In a classical job shop, operations have to be performed in a fixed sequence such that each operation has been pre-assigned a unique machine which may perform that operation so that a job does not visit the same machine twice.

- **Multi Stage, Open Shop Scheduling**

The open shop problem is the most general job scheduling problem in the classification; here there are no restrictions on the processing steps for a task, and alternative routings for a task may be allowed.

All the above mentioned scheduling problems could fall under the following cases:

a. ***Static and Dynamic:*** A static problem refers to one where the number of jobs and their ready times are known and fixed (French 1982). Problems in which jobs arrive randomly over a period of time are called dynamic. There exists many static cases in practice and experience from the static problem leads to understanding and insight into the dynamic problem. Also, the solution methods for the static sequencing problems help our understanding of the techniques in combinatorial optimization.

b. *Deterministic and Stochastic:* The deterministic case is the case where all elements of the problem, such as the state of arrival of the jobs in the shop, due dates of the jobs, processing times, availability of machines and so on do not include stochastic factors and are determined beforehand. The stochastic case refers to the case where at least one of those elements includes stochastic factors (Bellman, A.O. Esogbue and I. Nabeshima 1982).

1.1.2 Scheduling Criteria

Scheduling criteria indicate the measures with which schedules are evaluated. Two broad classes of criteria are schedule cost and schedule performance. A study of the criteria proposed in the literature indicates that a wide variety of measures of performance are employed. The most commonly used criteria are:

a. *Makespan:* The length of time required to complete all jobs is called the makespan (Thomas and David 1993).

b. *Average Job Flowtime:* A job flowtime is the amount of time between the job's arrival into and departure from the system (Thomas and David 1993). The average flowtime over all jobs in the schedule is used as a performance measure.

c. *Average Job Lateness:* A job lateness is the algebraic difference between the completion time and the due date of a particular job, regardless of the sign of the difference. The average lateness over all jobs in the schedule is used as a performance measure (Thomas and David 1993).

d. *Average Job Tardiness:* A job tardiness is the length of time taken to complete a particular job after its due date. Tardiness considers only the positive difference between the job completion time and its due date. The average tardiness over all jobs in the schedule is used as a performance measure (Thomas and David 1993).

e. ***Weighted Flowtime and Weighted Lateness:*** When using the mean flow time criterion, the jobs are given equal importance. One way to accommodate unequal importance is to assign a value or weighting factor to each job and to incorporate the weighting factors into the performance measure. These weights could be cost. The weighted version of mean flow time is called weighted mean flow time. The weighted version of lateness is called weighted lateness. The weighted version of tardiness is called weighted tardiness (Thomas and David 1993).

f. ***Maximum Lateness, Tardiness:*** Minimizing maximum tardiness is important when customers tolerate smaller tardiness but become rapidly and progressively more upset for larger ones. Minimizing maximum lateness is primarily important because it can be used as an aid for solving other problems (Thomas and David 1993).

g. ***Average Job Earliness:*** A job earliness is the length of time taken to complete a particular job ahead of its due date. Earliness considers only the negative difference between the job completion time and its due date. The average earliness over all jobs in the schedule is used as a performance measure.

In real situations, any one of the single criteria cannot be optimized at the expense of the others and thus, it is necessary and desirable to assess the alternate schedules with respect to multiple criteria. The most common way of defining multiple criteria is expressing different single criteria's into penalty and minimize the total penalty. Some of the multiple criteria studied in the literature (Fry, Arm strong and Black stone (1987), Fry, Leong and Rakes (1987b), Yano and Kim (1991, 1994), Davis and Kanet (1993), Lee and Kim (1995)) are given below:

a. ***Weighted Early-Tardy:*** This objective is used when the customer does not want jobs to be tardy but will not take early delivery. This kind of situation is very common in Just-In-Time manufacturing (Thomas and David 1993).

b. ***Weighted Early-Tardy-Flowtime:*** This objective is used to represent the conflicting objectives of inventory levels and customer satisfaction. Minimizing a mean flowtime to represent work-in-process inventory and total job earliness to represent finished goods inventory with total tardiness to represent customer satisfaction (Morton and Pentico 1993).

1.2 Mathematical Solution Methods

The theory of scheduling includes a variety of techniques that are useful in solving scheduling problems. Indeed, the scheduling field has become a focal point for the development, application, and evaluation of combinatorial procedures, simulation techniques, network methods, and heuristic solution approaches. The selection of an appropriate solution technique depends on the complexity of the problem, the nature of the model, the choice of a criterion, as well as other factors. In many cases it might be appropriate to consider several alternative techniques. Solution procedures for solving scheduling problems may be categorized as follows:

- **Optimization Procedures**

These optimization procedures provide optimal solutions for the given problem. The most common optimization procedures used in job scheduling problems are branch and bound which is basically an implicit enumeration technique that examines a subset of the feasible solutions, and dynamic programming which decomposes the problem into stages that are linked through recursive computations.

For both branch and bound and dynamic programming, while "small" problems can be solved exactly, large problems have remained intractable, and realistic problems are expected to remain so in the foreseeable future. The reason is that most scheduling problems are usually known to be "NP-complete", which means roughly that it is highly unlikely that a solution method can be discovered that doesn't grow exponentially as a

function of the problem size. For example, no computer is available that could check out all permutations of 50 jobs on one machine. Even if such an extraordinary machine were developed, to solve a problem with 55 jobs would require a computer about 300,000,000 times faster (Thomas 1993). Of course, exact mathematical methods are more clever than this, but running times do still go up exponentially in problem size in the same way

- **Heuristic Techniques**

Most of the problems in job sequencing are NP-complete. So it is very important to have efficient heuristic procedures to produce quickly near optimal solutions. The literature on heuristics is immense and we don't intend to mention all of it here. One class of heuristics are improvement procedures. Improvement heuristics begin with a feasible solution and successively improve it by a sequence of exchanges or mergers. Generally, the feasibility of the solution is maintained throughout the procedure. Some of the important improvement heuristic procedures are Tabu Search, Adjacent Pairwise Interchange, Pairwise Interchange, Beam Search, Genetic Algorithms, and Simulated Annealing. Other heuristics are construction algorithms that generate a solution by adding individual components one at a time until a feasible solution is obtained. 'Greedy' algorithms, seeking to maximize improvement at each step, comprise a large class of construction heuristics. In most construction heuristics, a feasible solution is not found until the end of the procedure. A classical example of a construction heuristic is the nearest neighbor heuristic for the Traveling Salesman Problem.

1.3 Thesis Problem Definition

This thesis mainly concentrates on static, deterministic, single machine scheduling problems where the objective is to minimize the total of earliness, tardiness, work-in-process, and machine idle costs. A special case of these problems is known to be NP-complete. So, we provide quick and efficient heuristic procedures to solve them. Beside

having their own intrinsic value, these problems can be used to solve more complex job shop scheduling problems. In particular, we consider the case where job have non zero release times and multiple due dates which we will refer to as the "decomposed job shop cost problem". As Amiouny (1995) shows, this problem can be used to solve job shop problems where the objective is to minimize total cost.

1.4 Organization of the Document

Chapter 2 presents an overview of the literature on minimizing cost in single machine scheduling problems. We found little work done on our problem of interest. We start by considering special cases of the decomposed job shop cost problem; This allows us to benchmark our heuristics against others found in the literature and to gain insight into the general case. Chapter 3 presents a formulation of the problem, the design of a heuristic procedure and experimental results for the special case where all jobs have zero release times and each job has a single due date. In chapter 4, we relax the assumption of zero release times and present a formulation of the problem, the design of another heuristic and experimental results. In chapter 5, we consider multiple due dates for each job and present a formulation of the problem, design and experimental analysis for the decomposed job shop cost problem. In chapter 6, we present our conclusions and suggest directions for future work.

CHAPTER 2
Literature Review

The study of earliness and tardiness (E/T) penalties in scheduling models has been a growing field of interest since the JIT philosophy started playing a major role in industry. The use of both earliness and tardiness penalties gives rise to a non regular performance measure i.e. the cost function may decrease as completion time increases, which leads to new methodological issues in the design of solution procedures. The vast majority of articles on *E/T* problems deal with deterministic single-machine scheduling models, although some single machine results have been extended to the parallel and stochastic case. Here, we review the literature on *E/T* problems and recent extensions to them.

2.1 Identical Earliness and Tardiness Penalties

In this model both earliness and tardiness are penalized at the same rate for all the jobs.

2.1.1 Common Due Date

This problem involves minimizing the sum of absolute deviations of the job completion times from a common due date *d*. This can be expressed as

$$TC = \sum_{i=1}^{n} |C_i - d|$$

This problem was examined by Kanet (1981 a), Sundararaghavan and Ahmed (1984), Hall (1986), Bagchi, Chang and Sullivan (1987), Hall, Kubiak and Sethi (1991), De, Ghosh, and Wells (1993), Lee (1994), and Ventura and Weng (1995). There are two possible assumptions about the problem. a). If '*d*' is too tight ($\sum_{i=1}^{n} p_i \leq d$), then it will not be possible to process all the jobs before '*d*' because no job can be processed before zero start time. b). If '*d*' is not tight ($\sum_{i=1}^{n} p_i > d$) then the problem is unrestricted. The

unrestricted problem has the following properties can be solved optimally in polynomial time.

I. There is no inserted idle time in the schedule.

II. The optimal schedule is V-shaped, i.e., jobs are sequenced in Largest Processing Time (LPT) order before the due date and in Shortest Processing Time (SPT) order after the due date.

III. One job completes precisely at the due date.

For the restricted version of the problem, only Properties I and II hold. A proof is given by Ragavachari (1986). Hall, Kubiak and Sethi (1989) have demonstrated that the restricted version of the problem is *NP*-complete. Bagchi, Sulivan and Cheng (1986) outline an algorithm for determining multiple optimal schedules for the restricted version assuming that the start time of the schedule is zero. Szwarc (1989) pointed out that the procedure does not give an optimal solution when the start time is non-zero. Baker and Chadowitz (1989) relax this condition and generalize the algorithm. Hall, Kubiak and Sethi (1991) solved a problem with common due date 'd', which is unrestrictively late by using dynamic programming. Later, Ventura and Weng (1995) were able to eliminate two subroutines from that pseudo-polynomial algorithm without affecting the final result. De, Ghosh and Wells (1993) examined the properties and concepts necessary for the general solution of two related classes of Early/Tardy problems, termed as $(ET)_r$ and $(WET)_r$ where r is a positive integer exponent of completion time variance from due date. They developed a solution methodology with a pseudo-polynomial complexity by using dynamic programming. They showed that this methodology applies effectively to instances of the two problem classes for $r \leq 2$. Lee (1994) developed simple and hybrid genetic algorithms by investigating basic operators for the applications of job sequencing problems. He presented experimental results by comparing his simple and hybrid heuristics against optimal solution procedure.

2.1.2 Distinct Due Dates

Oguz and Dincer (1994) examined the special case of single machine scheduling problem where the distinct due dates for the different jobs are related to processing times according to the equal-slack rule. After determining some properties of the problem, they showed the unrestricted case to be polynomial time solvable and the restricted case as to be *NP*-complete. The objective function is

$$TC = \sum_{i=1}^{n} |C_i - d_i|$$

where C_i is the completion time of job i and d_i is the due date of job i.

2.1.3 Non Linear Penalties

In some case, large deviations from the due date are highly undesirable. One way to account for this is to use the sum squared deviations from the common due date as the performance measure. The objective function is shown as

$$TC = \sum_{i=1}^{n} (C_i - d)^2 = \sum_{i=1}^{n} (E_i^2 + T_i^2)$$

where E_i is the earliness of job i and T_i is the tardiness of job i.

Bagchi, Sullivan and Chang (1987) show that the unrestricted version of this problem is equivalent to minimizing the sum of absolute variation of completion time to due date. This problem is also studied by Eilon and Chowdary (1977), Kanet (1981a), Vani and Ragavachari (1987), Weng and Ventura (1994). Bagchi, Chang and Sullivan (1987) also examine the general case in which earliness penalty is different from tardiness penalty assuming that the schedule starts at time zero. They proposed implicit enumeration procedures based on several dominance conditions and reported their computational experience. For the restricted version of the problem, De, Ghosh and Wells (1989a, b) present enumerative solution procedures without assuming that the schedule begins at time zero. Weng and Ventura (1994) presented an integer programming

formulation and a lagrangian solution procedure to approximately solve the tight restricted version of the problem.

2.1.4 Stochastic Models

The problem of scheduling n jobs on a single machine, which is subjected to random break downs, to minimize the expected sum of non regular penalty functions has been studied by John and Raghavachari (1993). They consider a simple resource model when the penalty function is the squared deviation of job completion times from a common due date. They developed characterizations of optimal schedules for the objective function when common due date is a decision variable and when it is given and fixed.

2.1.5 Parallel Machine Models

Hall (1986), Sudhararagavan and Ahmed (1984) extended the basic version of the unrestricted problem to the parallel machine case. They examined the minimization of total absolute deviation with m identical machines operating in parallel. Emmons (1987) extended this analysis to the case where earliness and tardiness are penalized at different cost rates. He also considered uniform parallel processors (non identical machines, where each machine has its own efficiency factor). He outlined an algorithm for the problem.

2.2 Different Earliness and Tardiness Penalties

Here all the jobs have identical penalties, but the earliness penalty is different from the tardiness one.

2.2.1 Common Due Date

This is a generalization of the previous model which has same earliness and tardiness penalties. The cost function to minimize is

$$TC = \sum_{i=1}^{n} (\alpha E_i + \beta T_i)$$

where α is the earliness penalty and β is the tardiness penalty.

This problem is analyzed by Bagchi, Chang and Sullivan (1987) and briefly by Emmons (1987), Krieger and Ragavachari (1992). The unrestricted version of the problem can be solved optimally by using properties I, II and III.

For the restricted version of the problem, which is *NP*-complete, Bagchi, and Chang and Sullivan (1987) showed that properties I and II hold. Baker and Chadowitz (1989) presented a heuristic solution approach for this version. Krieger and Ragavachari (1992) showed that any optimal schedule which minimizes the sum of the penalties (early or late) for all the jobs is V-shaped when the penalty function for each job is "monotone".

2.2.2 Different Due Dates

Szwarc (1993) examined single-machine *n* job earliness-tardiness model with job-independent penalties where processing time not necessarily starts at time zero. He demonstrated that the arrangement of adjacent jobs in an optimal schedule depends on critical value of the start times. Based on these precedence relations, he developed a criterion under which the problem can be decomposed into smaller problems. He claimed that the developed results can be incorporated in a branch-and-bound solution method. He mentioned that without the lower bound the branching scheme can handle only small problems. He tested 70 examples of the size n=10 where the branching scheme significantly reduced the search for an optimal schedule.

2.3 Job Dependent Earliness and Tardiness penalties

Here each job has independent earliness and tardiness penalties. This is generalized version of the problem mentioned in the previous section.

2.3.1 Common Due Date

In this case every job has its own penalty function, but all jobs have common due date. The cost function to minimize is

$$TC = \sum_{i=1}^{n} (\alpha_i E_i + \beta_i T_i)$$

The unrestricted version of this problem has been examined by Bagchi (1985), Cheng (1987), Quaddus (1987), Bector, and Gupta and Gupta (1988), Hoogeveen and Van De Velde (1991), De, Ghosh and Wells (1991), Bahram and Duane (1995), Lee and Kim (1995). The restricted version of this problem has been examined by Hall and Posner (1991), Hoogeveen, Oosterhout and Van De Velde (1994). The unrestricted version of the problem is *NP*-Hard. Bagchi (1985) considered the case in which the earliness penalties are proportional to processing times ($\alpha_i = \alpha p_i$). Bector, Gupta and Gupta (1988) presented a linear programming perspective on these same results. Hall and Posner (1991) established the relevant variations of properties I and II and also some dominance properties that provide necessary conditions for an optimal sequence. They proceeded to develop a dynamic programming algorithm, which they showed to be pseudo polynomial. Quaddus (1987) considered the general case in which $\alpha_i \neq \beta_i$, but only dealt with the selection of the common due date. But Baker and Scudder (1989) pointed out that Quaddus (1987) neglected the sequencing aspects of the problem. Hall and Posner (1991), Hoogeveen and Van De Velde (1991), De, Ghosh and Wells (1991) examined the case where $\alpha_i = \beta_i$ for all jobs. Hoogeveen and Van De Velde (1991) proved that this problem is *NP*-hard even if all job weights are equal. They presented a pseudo polynomial algorithm that requires O (n^2d) time and O (nd) space. De, Ghosh and Wells (1991) showed that the solution to this problem can be derived from solutions to different subproblems. Properties of these subproblems were identified and used to develop an exact algorithm of pseudo-polynomial complexity. Hogeveen, Oosterhout and Van De Velde (1994) developed a branch-and-bound algorithm based on lagrangian lower and

upper bounds that are found in O ($n\log n$) time. They proceeded to develop a 4/3-approximation based upon the lagrangian upper bound. Bahram and Duane (1995) showed that the V- shaped property holds for the general case. Lee and Kim (1995) developed parallel genetic algorithms for this problem. They illustrated the efficiency of the parallel genetic algorithm with computational results.

2.3.2 Distinct Due Dates

In this case earliness and tardiness penalties with independent job due dates are considered. The objective function for this case is same as the section 2.3.1.

Garey, Tarjan and Wilfong (1988) were the first to show that this problem is NP-complete. Recent papers exploring solution procedures include Yano and Kim (1991), Fry, and Armstrong and Blackstone (1987), Abdul-Razaq and Potts (1988), Szwarc (1988), Davis and Kanet (1993), Kim and Yano (1994), and Lee and Choi (1995). In this model Properties I and II do not hold. An optimal sequence may not be V-shaped and inserted idle time may be desired. The search for an optimal schedule can, however, be decomposed into two subproblems as follows: Finding a good job sequence; and scheduling inserted idle times.

Abdul-Razaq and Potts (1988) use a relaxed dynamic programming procedure to obtain good bounds. But they did not insert any idle time between successive operations. Szwarc (1988) gives a branching procedure for finding an optimal schedule without inserting idle time. But he did not report any computational experience. Ow and Morton (1988,1989) examined a few heuristics without inserting idle time between operations. Garey, Tarjan and Wilfong (1988) gave the procedure to insert idle times optimally for symmetrical earliness and tardiness penalties. The proposed algorithm runs in O ($n\log n$) time. Fry, Armstrong and Blackstone (1987a) described a linear programming procedure to insert idle times optimally for a given sequence. In their model, asymmetrical earliness and tardiness penalties are assumed. Davis and Kanet (1993) presented a time tabular

procedure to insert idle times for a given sequence. They claimed that time tabular procedure takes modest time to get optimal start times. They presented computational experience for few heuristic procedures. Kim and Yano (1994) identified several properties of optimal solutions for this problem. They developed a branch-and-bound algorithm and a heuristic algorithm (priority relationships) based on these properties. They observed that CPU time requirements for the branch-and-bound procedure depend heavily on the range of due dates. That is, problems with less widely dispersed due dates are more difficult to solve optimally with the branch-and-bound algorithm. Lee and Choi (1995) developed a genetic algorithm by employing problem specific reproduction, crossover and mutation operators.

2.4 Due Date as a Decision Variable

Here we review the problem of finding an optimal schedule and due date for a set of jobs, in a static single machine system. The scheduling criterion is the minimization of the earliness and tardiness penalties. This problem has been studied by Cheng (1985, 1986a, b, 1988a, b, c, 1990), Quaddus (1987a, b), Bector (1987, 1988), Bector, Guptha and Guptha (1988), Suresh and Dilip (1992), and Parthasarati (1993). Cheng (1989) presented a review article where due date as a decision variable. Cheng (1985) outlines a procedure to find an optimal due date by solving the linear programming dual problem to minimize weighted average of the absolute value of job lateness. This problem has been extended to determine the optimal value of the flow allowance (which is a constant added to the processing time to decide due date) and the optimal job sequence to minimize a cost function based on the flow allowance and the job earliness and tardiness values by Cheng (1988b) where he solved the problem by using linear programming. Cheng (1988a) studied the problem of finding the optimal common flow allowance for the common due date assignment method, and the objective function is based on the idea that there is no penalty for a job that misses its due date marginally. When a job misses its due

date by a big margin, it assumed that the penalty is proportional to the amount of missed due dates, regardless of its being early or tardy. He presented three lemmas to enable the determination of the optimal due-date and sequence.

Quaddus (1987a) developed a linear-programming model to find the optimal 'CON due-date' i.e. constant-allowance due date, where each job receives exactly the same due date. He considered a more generalized version (job dependent) of minimizing earliness and tardiness problem and presented an optimal solution procedure by using duality theory. Quaddus (1987b) considered the problem of minimizing absolute deviation of completion times from common due date and presented a linear programming procedure to find an optimal CON due date which is solved by considering its dual. Cheng (1990) presented a partial search algorithm to solve single-machine common due-date assignment and sequencing problems. This partial search algorithm is not polynomial-bound, but he claimed that it is showing better performance in terms of computational efficiency than that presented in Cheng (1987).

Bector, Gupta and Gupta (1988) developed an optimal algorithm based on the idea of sensitivity analysis in linear programming and elementary concepts in linear goal programming to find an optimal value of common due date and a corresponding optimal sequence for minimizing mean absolute deviation of completion times from a common due date. Suresh and Dilip (1992) developed an optimal algorithm of order O ($n\log n$) to determine optimal due dates and a sequence for an n-job problem when the number of jobs to be assigned to different due dates. They developed important dominance property to determine the number of jobs to be assigned to different due dates. Parthasarati (1993) presented a branching procedure for finding an optimal solution to a common due date case to minimize weighted sum of earliness and tardiness penalties.

2.5 General E/T Models

Liman and Ramaswamy (1994) studied the problem of n jobs on single-machine to minimize the sum of weighted earliness and weighted number of tardy jobs given a delivery window. There is no penalty if jobs are completed with in the delivery window. They assumed that the length of delivery window (which corresponds to the time period within which the customer is willing to take deliveries) is a given constant. They considered two cases of the problem: a). Unrestricted window case (one where the position of the delivery window is internally determined). b). Restricted window case (one where the position of the delivery window is externally specified parameter). They described some optimal properties, and proved that the problem is *NP*-complete even in the unrestricted window case. They presented dynamic programming algorithms for both cases. They did not report any computational experience.

Li, Cheng and Chen (1995) considered the objective of minimizing the weighted number of early and tardy jobs under the condition that the start times and due dates are "agreeable", i.e., the start times must increase in the same sequence as the due dates. They showed that the problem is *NP*-hard in the strong sense. They discussed the computational complexity of the case when all earliness penalties are the same and all tardiness penalties are same. They proposed a dynamic programming approach to solve a special case of the E/T problem in pseudo-polynomial time. They presented two heuristics to deal with the general problem and claimed that one of the heuristics will generate near optimal solutions. They presented some computational results and suggested directions for future research.

2.6 FET Model

The *E/T* problem was extended by Fry, Leong and Rakes (1987b) to FET (Flow time, Earliness and Tardiness penalties), where they included work in process penalty in the model. They considered independent job earliness and tardiness and work in process

penalties. They studied the static, single-machine, linear cost case by inserting idle time in-between job operations and described a linear programming procedure to insert idle time optimally between jobs. The objective function they used can be expressed as

$$TC = \sum_{i=1}^{n} (\gamma_i W_i + \alpha_i E_i + \beta_i T_i)$$

Fry, Leong and Rakes (1987b) examined two procedures for the above problem. The first is an enumeration scheme using bounding and dominance criteria and the second is a mixed integer linear programming formulation. Computational experience with the two models was also presented.

2.6.1 Non-linear FET Problem

Zheng, Nagasawa and Nishiyama (1993) examined an unconstrained single-machine scheduling problem with a common due date where the objective is to minimize the total cost of flow time, earliness and tardiness. They considered a non-linear cost function that is identical for all the jobs. They proposed several dominance conditions necessary for an optimal schedule. However Weng and Ventura (1994) point out that two key dominant properties that were derived by Zheng, Nagasawa and Nishiyama (1993) do not necessarily hold for an optimal sequence.

2.7 Other Models

Lee, Danusaputro and Lin (1991) examined the *n*-job, non-preemptive, single-machine problem of minimizing the weighted sum of tardy jobs and earliness-tardiness penalties about a common due date. They studied the case of job-dependent penalties under a certain condition on the ratio of processing times and weights. They considered two cases: a) Due date as a given parameter. b) Due date as a decision variable, and provided dynamic programming algorithms to solve both instances. They proposed some efficient algorithms for the special case when the weights are independent of jobs.

Sung and Joo (1992) examined the single-machine scheduling problem with a common due date. The objective function is to minimize the sum of earliness, tardiness and starting-time penalties in the situation where all jobs are not required to be available at time zero. They characterized several dominant solution properties with respect to assigning either early or tardy positions to each job and to sub-sequencing all the jobs assigned in such early and tardy sub-sets, respectively. They suggested a stepwise procedure for finding an optimal starting time for an arbitrary given sequence. These are then all put together to construct a heuristic algorithm for a good solution search. They presented their computational experience with the proposed heuristic algorithm.

Herrmann and Lee (1993) examined the single-machine problem of minimizing the sum of earliness and tardiness penalties and the delivery costs of the tardy jobs, where the tardy jobs are delivered in batches with a fixed cost per batch about a common due date. They used a pseudo-polynomial dynamic programming algorithm to solve the problem. They also discussed some special cases that are solvable in polynomial time and showed that for a given schedule of tardy jobs, the problem of scheduling the batch deliveries is equivalent to the dynamic lot sizing problem. Finally, they presented the results of empirical testing of the dynamic program and a number of heuristics developed.

2.8 Concluding Remarks

The above research does not include machine idle cost into the model, although we believe it is an important component. An industrial survey conducted by the sequencing research group at Texas Tech University indicated that the percentage idleness of the machines varies from 15% to 35%. This survey shows the need of including machine idle cost in the objective function (Gupta 1970). Moreover, keeping machine idle may cause future delays. Adil, Rajamani and Strong (1993) developed a mathematical model for cell formation by considering machine idle time penalty, work in process penalty, earliness penalty and tardiness penalty. This kind of formulation for

scheduling problems is very important to generalize the model. A lot of work needs to be done towards this kind of problems to get efficient solution methods that find near optimal solution because this problem is *NP*-complete. More computational experience is required to find a good heuristics for this model. Another aspect of the decomposed job shop cost problem that is not addressed in the literature is multiple due dates and multiple tardiness penalties for each job.

CHAPTER 3

Single due date per job and equal release times

In this chapter, we formulate a model and design a new heuristic procedure for single machine scheduling problems in which jobs have zero release times and the objective is to minimize the total of earliness, tardiness, work-in-process, and machine idle costs. We also propose a new heuristic procedure to insert idle times optimally for a given sequence. We start with this simplified version of the problem for several reasons:

- We believe that this problem provides good understanding and foundation to develop efficient heuristic procedures to solve the decomposed job shop cost problem.
- Simplified problems often allow to gain insight into the more sophisticated ones. In particular, we wanted to identify the factors that had the most effect on the relative performance of different heuristics.
- Other authors (Fry, Armstrong, and Blackstone (1987a), Yano and Kim (1991), Davis, Kanet (1993)) looked at very similar problems. This allow us to have a benchmark to compare our results.
- The problem is of interest in itself as the interest it has generated proves.

3.1 Problem Formulation

Consider a non-preemptive, single-machine scheduling problem with n jobs. All jobs with deterministic processing times p_i have possibly different due dates d_i and are ready for processing at time zero. Set up times are included in processing times. Idle time is allowed between any successive job operations. This model includes work-in-process cost, earliness penalty cost, tardiness penalty cost, and machine idle cost. Here

the objective is to minimize the total cost consisting of the sum of the above mentioned costs:

Total Cost = Work-In-Process Cost + Earliness Penalty Cost +
Tardiness Penalty Cost + Machine Idle Cost.

Work-In-Process Cost (WPC) : This cost, which is also known as the flowtime cost, is the cost incurred due to the wait for processing of unfinished jobs in the shop. The job which waits in the shop is a form of capital tied up in the shop. This capital could have been utilized to produce additional return on capital. For a job i with work in process penalty γ_i, and flowtime (or completion time) C_i, we assume that the penalty is a linear function of time:

$$\text{WPC} = \gamma_i C_i$$

Earliness Penalty Cost (EPC) : This cost is due to early completion of jobs. Customers are often unwilling to receive goods before a certain date (the due date) due to the JIT manufacturing philosophy. Therefore, a job that finishes early incurs a cost for finished goods inventory. Other than that, early job completion can cause the cash commitment to resources in a time frame earlier than needed, giving rise to early completion penalties. For a job i with earliness penalty α_i and earliness E_i, We assume that the penalty is a linear function of time:

$$\text{EPC} = \alpha_i E_i$$

Tardiness Penalty Cost (TPC) : This cost is due to late completion (after the due date) incurred due to various reasons, such as loss of good will, penalty clause in the contract, or opportunity costs of lost sales. For a job i with tardiness penalty β_i and tardiness T_i, We assume that the penalty is a linear function of time:

$$\text{TPC} = \beta_i T_i$$

Machine Idle Cost (MIC) : This is the cost incurred for keeping the machine idle. Due to the JIT production, jobs need to be processed on time which force the machine to be idle. If a machine is kept idle to schedule present jobs effectively, it may cause jobs waiting for machine in the future. Depending upon the expected future demand, it may be wise to give some penalty for keeping the machine idle. Where machine idle cost per unit time is μ and M_t is machine idle time, that cost is

$$\text{MIC} = \mu M_t$$

We represent single machine scheduling problem with the objective of minimizing the total of work-in-process, earliness, tardiness, and machine idle costs as $n/1/FETM$. The function we seek to minimize is the sum of all those costs, and can be expressed as

$$\text{TC} = \sum_{i=1}^{n} (\gamma_i W_i + \alpha_i E_i + \beta_i T_i) + \mu M_t$$

The above problem is *NP*-complete in the strong sense, since it is a generalization of the weighted tardiness problem, which is known to NP-hard in the strong sense (Lenstra 1977, Lawler 1977). Fry, Armstrong, and Blackstone (1987a), Yano and Kim (1991), Davis, Kanet (1993) examined *E/T* problem as two different problems. First they find best schedule without inserting idle times in between jobs and then they insert idle times optimally for the obtained schedule. If we follow that procedure we need to have an algorithm which inserts idle times optimally for $n/1/FETM$ problem.

3.2 An Optimal Idle Time Insertion Algorithm

Here we present an algorithm to insert idle times optimally in between job operations for $n/1/FETM$ problem. Several authors have suggested optimal

procedures to insert idle times given the sequence. Here we developed another such procedure that we feel is more elegant and easier to understand and implement.

Our procedure starts with all the jobs scheduled, in the given sequence, starting at time zero and with no inserted idle time. Starting with the first job, we insert idle time before a job j only if moving that job and all succeeding ones forward by one time unit results in a reduction in total cost. The amount of idle time we insert is equal to the minimum non-zero earliness among j and its successors, since the cost reduction will not change over that range when all costs are linear. Then we recalculate the potential cost reduction to see whether inserting more idle time is beneficial. Denoting by C_i, E_i, and T_i the completion time, earliness and tardiness of job i, the above procedure is formally stated as follows:

Algorithm IDLE TIME INSERT [ITI]: Schedules n jobs in a given sequence to minimize the total of flowtime, earliness, tardiness, and machine idle costs.

1. Number the jobs 1,..., n in the order of the given sequence, from a schedule with no inserted idle times, and compute their earliness and tardiness:
   (a) Set $C_1 = p_1$, $E_1 = \max\{0, d_1 - C_1\}$, and $T_1 = \max\{0, C_1 - d_1\}$;
   (b) Set $C_i = C_{i-1} + p_i$, $E_i = \max\{0, d_i - C_i\}$, and $T_i = \max\{0, C_i - d_i\}$ for $i = 2,\ldots,n$.
2. Initialize two counters $j \leftarrow 1$, $k \leftarrow 1$, and a number to keep track of potential cost reduction from inserted idle time $CR \leftarrow 0$.
3. Compute the change in cost when pushing job k forward one time unit, and add that cost to CR:

$$CR \leftarrow CR + \alpha_k - \gamma_k \qquad \text{if } E_k > 0$$

$$CR \leftarrow CR - \beta_k - \gamma_k \qquad \text{otherwise}$$

4. Check the values of CR and k:

(a) If $CR > 0$ and $k < n$, set $k \leftarrow k+1$ and go to step 3.

(b) If $CR \leq 0$ and $k < n$, set $j \leftarrow k+1$, $k \leftarrow k+1$, $CR \leftarrow 0$ and go to step 3.

(c) If $CR > \mu$ and $k = n$, move all the jobs $j,\ldots, n$ forward in the time by $\min_{j \le i \le n} E_i$. Update the earliness and tardiness of the moved jobs, set $k \leftarrow j$, $CR \leftarrow 0$ and go to step 3.

(d) If $CR \leq \mu$ and $k = n$, stop.

In step 4(b), since the cost increases by moving jobs $j,\ldots, k$ forward, we stop considering those jobs for idle insertion and continue to start with job $k+1$. Note that each time step 4(c) applies, at least one job that was early becomes on-time. Since there can be no further reduction in cost when all jobs are either on-time or tardy, there will be at most n iterations of the algorithm. In each iteration, there will be at most n jobs to consider, so the time complexity of the IDLE TIME INSERT is $O(n^2)$.

Theorem 1 *For a given sequence of jobs, algorithm "*IDLE TIME INSERT*" inserts idle times optimally in between jobs when all costs are linear.*

Proof: Denote by σ the schedule obtained from Optimal timing algorithm and let σ' be any other schedule with the same sequence but at least one difference in inserted idle times. Let j be the first job such that the idle I_j before j in σ is different from the idle time I_j' before j in σ'.

If $I_j > I_j'$, we know that moving j and any number of succeeding jobs forward by one time unit will reduce the total cost, so σ' can not be optimal.

If $I_j < I_j'$, let k denote the first job after j that has idle time after it in σ. We know that moving the jobs $i,\ldots,k$ for any $j \le i \le k$ forward by one time unit will not reduce the total cost. Now let k' denote the first job after j that has idle time after it in σ'. If $k' \ge k$, then moving the jobs $j,\ldots k$ in σ' backward in time by one time unit will not make σ' any worse. Otherwise, moving jobs $k',\ldots, k$ in σ' backward in time by

one time unit will not make σ' any worse. Therefore, σ will be no worse by at least one of the above reductions in idle time. ❑

3.3 Dominance Conditions

One way of reducing the enumeration is through the establishment of dominance relations. Such dominance conditions are established here for $n/1/FETM$ problem. The possibility of inserted idle time complicates the problem since a job which was once early in a partial sequence may be forced tardy as more jobs are added to that partial sequence and idle time is inserted.

Lemma 1: *In an optimal schedule* σ^*, *all jobs scheduled after the latest due date must appear in non-increasing order of the ratio* $(\beta_i + \gamma_i)/p_i$.

Proof: The proof is by a simple interchange argument. First it is obvious that σ should contain no idle time between any two jobs scheduled after the latest due date. Now assume that there are two adjacent jobs i and j such that both i and j start after the latest due date, i is scheduled before j, $(\beta_i+\gamma_i)/p_i < (\beta_j+\gamma_j)/p_j$, and there is no idle time between i and j. Interchanging i and j results in a change in total cost of

$$
\begin{aligned}
& (\beta_i+\gamma_i)(C_i'-C_i)+(\beta_j+\gamma_j)(C_j'-C_i') \\
= \; & (\beta_i+\gamma_i)(C_i'+p_j-C_i')+(\beta_j+\gamma_j)(C_i'-p_i-C_i') \\
= \; & (\beta_i+\gamma_i)(p_j)-(\beta_j+\gamma_j)(p_i) \\
< \; & 0 \text{ since } \frac{\beta_i+\gamma_i}{p_i} < \frac{\beta_j+\gamma_j}{p_j}
\end{aligned}
$$

So interchanging i and j reduces the total cost. ❑

Lemma 2: *In an optimal schedule* σ^*, *all jobs scheduled before the earliest due date must appear in non-decreasing order of the ratio* $(\alpha_i-\gamma_i)/p_i$.

Proof: Same as before. ❑

Lemma 3: *Consider a sequence with adjacent jobs i and j such that job i is late, job j is early and* $i \leftarrow j$. *Interchanging i and j will never decrease the penalty unless* $p_i(\gamma_j - \alpha_j) > p_j(\beta_i + \gamma_i)$.

Proof: The interchange will increase the lateness of the late job and the earliness of the early job. Therefore cost will never decrease unless $p_i(\gamma_j - \alpha_j) > p_j(\beta_i + \gamma_i)$ is true. ❑

3.4 A Job Insertion Heuristic

Since $n/1/ETM$ scheduling problem is known to be an NP-complete, the development of heuristic algorithms that guarantee good solution becomes necessary to solve big problem sizes, while optimal procedures can solve very small problem sizes. Existing heuristics for earliness tardiness problems are mostly applications of general search techniques. Here we develop an algorithm that is specially tailored to the problem and takes advantage of the lemma 1.

We start by sorting jobs in non-increasing order of $(\beta_i + \gamma_i)/p_i$ and try to schedule each job at its due date. If it does not overlap any previously scheduled jobs, we go on to the next one. Otherwise, we try placing that job at the beginning, then at the end, and then we try three ways of inserting it in between previously placed jobs close to its due date, pushing other jobs earlier or later so that it fits. Finally we keep it in the position that results in the lowest total cost.

To make the presentation easier, we use the expression "pushing jobs forward" for a set of scheduled jobs to designate changing the start times $C_i - p_i$ of those jobs from their current values to larger values while keeping them in the same sequence and satisfying the constraints that their processing time intervals can not overlap. We also use the expression "t is in currently idle time" for a point in time t to mean that in the current schedule, there is no job i such that $C_i - p_i \le t \le C_i$. The details are as follows:

Algorithm GREEDY INSERT [GI]: Schedules n jobs on a single machine in an attempt to minimize the total of flowtime, earliness, tardiness, and machine idle costs, which need not be linear.

1. Sort the jobs in non-increasing order of $(\beta_i + \gamma_i)/p_i$. Break ties by earliest due date first.
2. For $i = 1, 2, \ldots, n$ do:
   - If placing i so that it completes exactly at its due-date does not interfere with any of the previously placed jobs, place it there.
   - Otherwise, select among the following five schedules the one with lowest total cost:

   (a) Place i so that it comes first in the sequence of scheduled jobs, with no idle time between i and the next job. This may require pushing forward some previously scheduled jobs.

   (b) Place i so that it comes last in the sequence of scheduled jobs, with no idle time between i and the preceding job.

   (c) If d_i is in currently idle time, place i so that it completes at d_i and push preceding jobs earlier so that they do not overlap. Otherwise, find the closest idle time that occurs prior to d_i and insert i there, pushing preceding jobs earlier if necessary. In both cases, if pushing preceding jobs earlier is not enough, succeeding jobs are pushed later.

   (d) If $d_i - p_i$ is in currently idle time, place i in that idle interval as late as possible if it fits, or by pushing succeeding jobs later if it does not. Otherwise, find the closest idle time that occurs after $d_i - p_i$ and insert i there, pushing succeeding jobs later if necessary.

   (e) If d_i is in currently idle time and job i fits in that idle interval, place it so that it starts at the start of the idle interval. If it does not fit, place it so that it

completes at the end of the idle interval and push preceding jobs earlier to make it fit.

The five schedules tested in step 2 may not all be distinct ones. However, at least two options are tested: One placing the job early, and one placing it tardy. So although we are initially sorting the jobs according to their tardiness and flowtime penalties only, the effect of the earliness penalties and of the machine idle cost comes to play when choosing among the five schedules of step 2.

GREEDY INSERT works in $O(n^2)$ and results in a schedule with idle times already inserted, through not necessarily in an optimal fashion. This is important especially when the costs involved are not linear, since the IDLE TIME INSERT algorithm and other similar procedures presented elsewhere require all costs to be linear.

Numerical Example Illustrating GREEDY INSERT

Consider the $n/1/FETM$ problem with 5 jobs, specified by the data in Table 3.1.

Table 3.1 Example Scheduling Problem:

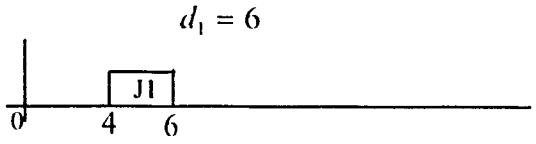
Jobs	Processing Time	Due date	Earliness Penalty	Tardiness Penalty	Flowtime Penalty
1	2	6	6	10	1
2	5	18	4	8	1
3	8	15	10	20	2
4	10	30	8	13	4
5	4	4	6	8	2

Machine Idle cost = 1

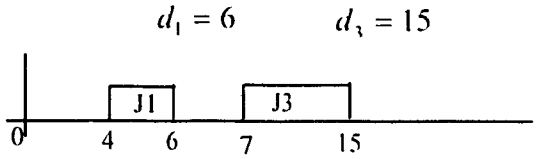
Solution:

Step 1: If we sort jobs by non - increasing order of $(\beta_i + \gamma_i)/p_i$, we get a sequence of 1-3-5-2-4.

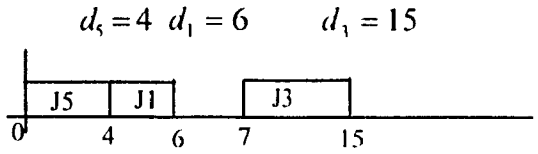
Step 2: Since Job 1 is a first job, place it exactly at its due date.



Placing Job 3: completes exactly at its due date and does not interfere with any of the placed jobs, so place it there.

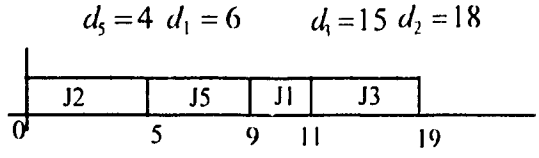


Placing Job 5: completes exactly at its due date and does not interfere with any of the placed jobs, so place it there.



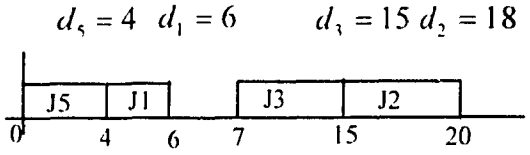
Placing Job 2: at its due date will interfere with job 3. So, following 5 cases will be examined and schedule with least cost will be chosen.

(a). Place job 2 so that it comes first in the sequence of scheduled jobs, with no idle time between job 2 and the next job. This may require pushing forward some of the previously scheduled jobs such as J5, J1, J3.



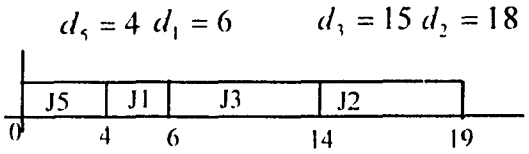
Total cost for case A = 294

(b). Place job 2 so that it comes last in the sequence of scheduled jobs, with no idle time between job 2 and the preceding job 3.



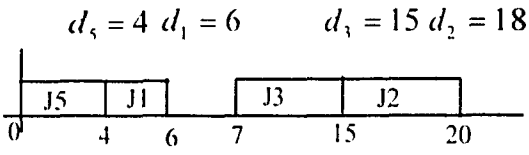
Total cost for case B = 81

(c). Since due date of job 2 is in currently idle time, place at its due date 18 and try to push all the previous jobs earlier so that they do not overlap.



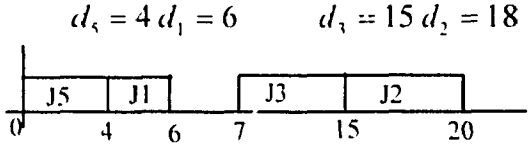
Total cost for case C = 79

(d). Since $d_2 - p_2$ interfere with job J3, we find the closest idle time that occurs after $d_2 - p_2$ and then insert J2 there, pushing succeeding jobs later if necessary.



Total cost for case D = 81

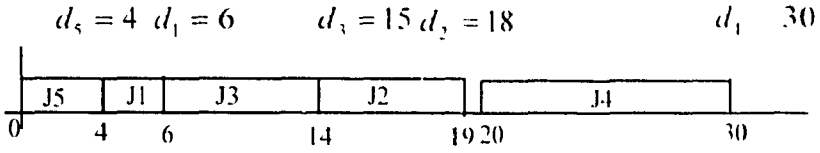
(e). Since the due date of job 2 in the idle time and fits in that interval, we schedule job 2 to start at time at the start of the idle interval i.e. 15. The schedule looks as shown below.



Total cost of case E=81

From the above select the schedule with lowest cost i.e. case (C).

Placing Job 4: completes exactly at its due date and does not interfere with any of the placed jobs, so place it there.



Total cost for this schedule =200

3.5 Heuristics Used for Comparison

The computational effort required to exactly solve NP-hard problems grows remarkably fast as the problem size increases. Comparison of GREEDY INSERT with optimal solutions for large problems (greater than 15 jobs) is very difficult, so we restrict the comparison to some of the efficient search techniques that were found to perform well elsewhere. We present a brief description of such procedures, the first four of which are dispatching rules, and the last three are neighborhood search techniques.

SHORTEST PROCESSING TIME [SPT]: The jobs are sequenced in non-decreasing order of their processing time. Ties are broken by earliest due-date first. We consider this dispatching rule since our total cost includes work-in-process cost and it is known that [SPT] minimizes flow time in single machine scheduling problems.

EARLIEST DUE DATE [EDD]: The jobs are sequenced in non-decreasing order of their due-date. Ties are broken by shortest processing time first. We consider this

dispatching rule since our total cost includes earliness, tardiness costs and it is known that [EDD] minimizes the maximum job lateness and the maximum job tardiness in single machine scheduling problems.

SHORTEST SLACK [SLK]: The jobs are sequenced in non-decreasing order of their slack (for a given job is the until due date minus the time required to process it). Ties are broken by earliest due date first. We consider this dispatching rule since many researchers felt that [SLK] provides reasonably good solutions in less time.

BEST OF EDD AND SLK [E/S]: This selects best solution obtained by EDD and SLK for a given problem. We consider this dispatching rule since it provides always best solution out of [EDD] and [SLK] in less time.

ADJACENT PAIRWISE INTERCHANGE: This is a search algorithm that starts with a given sequence (in our experiments, we used the solution from [EDD]) and tries to improve on it by interchanging the positions of adjacent jobs in the sequence. We consider two implementations of the search:

1. API1 begins at the first position in sequence and proceeds front to back. When interchanging a pair of adjacent jobs results in cost reduction, the interchange is made, and the search for a favorable interchange begins from that position onwards in the sequence. This procedure terminates when their is no improvement found in (n-1) comparisons from front to back.

2. API2 forms the (n-1) sequences obtained by a single adjacent pairwise interchange operation (the "neighborhood") and evaluates the total cost of each, then takes the one which results in the least cost. This procedure is repeated on the last sequence until their is no further improvement in the total cost.

PAIRWISE INTERCHANGE: This search algorithm is similar to the previous one except that it considers the interchange of any two jobs in the sequence. Again, we consider two implementations of the search:

1. PI1 : This algorithm starts with the initial sequence (in our experiments, we used the solution from [EDD]) and tries to improve the solution by considering all possible pairwise interchanges. Unlike API1, the jobs need not be adjacent. Here we do immediate interchange if it results in any reduction of total cost before considering the next pair. This procedure terminates when their is no improvement found in $n(n\text{-}1)/2$ comparisons from front to back.

2. PI2 : Unlike the above here we form the neighborhood for the solution available form [EDD] by pairwise interchange operation and evaluate the best possible seed in the neighborhood to be consider as a next sequence. Here we form $n(n\text{-}1)/2$ sequences as a neighborhood for each step. This procedure continues until there is no improvement in by forming the neighborhood.

TABU SEARCH: Tabu search, which is a strategy designed to guide other methods (in this case search methods) to escape the trap of local optimality for solving optimization problems, saw its beginning over a decade ago and was first fully described in by Fred Glover (1977, 1986, 1990). Since that time, tabu search has been shown to be a remarkably effective approach to a wide spectrum of problems. Tabu search has obtained optimal and near optimal solutions to a wide variety of classical and practical problems in applications ranging from scheduling to telecommunications and from character recognition to neural networks (Fred Glover 1990). This search algorithm keeps a list of recently visited sequences (the tabu list) and moves in each iteration to the best sequence that is in the neighborhood of the current one but not in the tabu list, even if that sequence results in an increase in cost. This allows for diversification in the

search and may prevent it from being stuck at a locally optimal sequence. In our implementation, we define the neighborhood as consisting of all possible pairwise interchanges of a sequence. When all moves in the neighborhood are in the tabu list, we choose randomly among them in order to continue the search (this is known as the aspiration criterion). We stop the search after a fixed number of iterations, and use the notation [TABUk] to denote a search with k iterations.

3.6 Experimental Design

In order to test the performance of our heuristic and compare it to other procedures, we designed a 2^5 factorial experiment that takes into account the following factors:

1. The tardiness factor T, which is a measure of the tightness of the due dates.
2. The due date range R, which is a measure of how spread out the due dates are around their average.
3. The earliness cost to tardiness cost ratio.
4. The flowtime cost to earliness cost ratio.
5. The machine idle cost.

The first four factors are used by other researchers, we added machine idle cost in generating problems to see the effect on the performance of heuristics. The full factorial design to conduct experiment is shown in the Table 3.2. In order to be consistent with results reported elsewhere, we randomly generated the processing times of the jobs from a Discrete Uniform [1,30] distribution, the due dates from a Discrete Uniform $[P(1- T - R/2), P(1 - T + R/2)]$ distribution where $P = \sum_{i=1}^{n} p_i$, and the tardiness costs per unit time β_i from a Continuous Uniform[1,5] distribution. We ran the experiment for instances with 10, 20, 30, 40, and 50 jobs per problem and tested GREEDY INSERT against heuristics that have been proposed by others for solving earliness tardiness problems with different due dates, and that can be easily modified to

accommodate flowtime and machine idle costs. All the algorithms presented in this chapter were coded in PASCAL and run on Pentium at 66 MHz.

Table 3.2 Experimental design to calculate the effect of different factors on the performance of heuristics:

Design Number	Factors considered in the experiment				
	T	R	α / β	γ / α	μ
1	0.1	0.8	0.25	0.1	5.0
2	0.1	0.8	0.25	0.1	25.0
3	0.1	0.8	0.25	0.25	5.0
4	0.1	0.8	0.25	0.25	25.0
5	0.1	0.8	0.75	0.1	5.0
6	0.1	0.8	0.75	0.1	25.0
7	0.1	0.8	0.75	0.25	5.0
8	0.1	0.8	0.75	0.25	25.0
9	0.1	1.2	0.25	0.1	5.0
10	0.1	1.2	0.25	0.1	25.0
11	0.1	1.2	0.25	0.25	5.0
12	0.1	1.2	0.25	0.25	25.0
13	0.1	1.2	0.75	0.1	5.0
14	0.1	1.2	0.75	0.1	25.0
15	0.1	1.2	0.75	0.25	5.0
16	0.1	1.2	0.75	0.25	25.0
17	0.4	0.8	0.25	0.1	5.0
18	0.4	0.8	0.25	0.1	25.0
19	0.4	0.8	0.25	0.25	5.0
20	0.4	0.8	0.25	0.25	25.0
21	0.4	0.8	0.75	0.1	5.0
22	0.4	0.8	0.75	0.1	25.0
23	0.4	0.8	0.75	0.25	5.0
24	0.4	0.8	0.75	0.25	25.0
25	0.4	1.2	0.25	0.1	5.0
26	0.4	1.2	0.25	0.1	25.0
27	0.4	1.2	0.25	0.25	5.0
28	0.4	1.2	0.25	0.25	25.0
29	0.4	1.2	0.75	0.1	5.0
30	0.4	1.2	0.75	0.1	25.0
31	0.4	1.2	0.75	0.25	5.0
32	0.4	1.2	0.75	0.25	25.0

3.7 Experimental Results

For [SPT], [EDD], and [SLK], we used an IDLE TIME INSERT on the sequence obtained. For the search procedures, we first tested their performance when the improvement is judged based on sequences with no inserted idle time, and used IDLE TIME INSERT algorithm only when search concluded. This performed very poorly. In many cases, the cost increased with respect to the initial sequence with idle times inserted optimally. So in all the presentation that follows, the searches are done by inserting idle times optimally for each member of the neighborhood.

In a first phase, we ran a few cases to determine the number of runs required for the average performance to converge. We settled on 200 runs for each case which gave us a confidence interval of about ±3% around the average performance. So, we generated and tested a total of 6400 problems for each value of n (200 for each design). For convenience, we present in Table 3.3 the average performance over the 200 problems of each heuristic relative to the average performance of [API1]. To see the individual effects of the five factors Table 3.4 presents the average performance of the heuristics relative to [API1] for each factor separately. For example, in Table 3.4(a), each entry corresponds to the average of 3200 problems corresponding to all designs where T is either high or low. From Table 3.4, it is clear that the most significant factors affecting the relative performance of [GI] are T and R. Accordingly, we will concentrate from now those two factors only. Figures 3.1 plots the results as the ratio of the performance of [API1] to that of the heuristic for different values of T and R. These were averaged over the performances at eight combinations of levels of the other three factors. We used [GI+] to designate the performance of GREEDY INSERT followed by the application of the IDLE TIME INSERT algorithm. The exact average total cost figures obtained from running problems for all heuristics at different T and R values are tabulated in Appendix-A which includes Tables A.1, A.2, A.3, A.4. Average CPU times are tabulated in A.5.

Some interesting results of our study are listed below, where we use "significant" to denote 90% confidence level significance:

- There was an insignificant difference in average performance between [API1] and [API2], and between [PI1] and [PI2], but a substantial increase in running time. [PI1] however was significantly better than [API1], and more so for a larger number of jobs, but at the expense of a substantial increase in running time (about 6 times more for 20 jobs and 25 times more for 50 jobs).

- While the results of all search procedures plotted in Figure 3.1 and in all tables are for an [EDD] starting sequence, we have tested several starting sequences. We could only detect a significant difference in performance for [API1] and [API2] where a better starting sequence resulted in better performance. For [PI1], [PI2] and [TABU 100] however, the starting sequence was practically irrelevant.

- While [SLK] performs significantly better that [EDD] for loose due dates (T = 0.1), [EDD] seems to be doing better for tight due dates (T = 0.4).

- Performance of Tabu Search is greater than all other heuristics presented, but it is very expensive in terms of computation.

- GREEDY INSERT runs in a fraction of a time of the search heuristics, but performs very well on average, even better than [API1] for small range R.

Table 3.3(a) Average Performance of heuristics relative to API1 for 10 Jobs:

Due dates	Range	E/T	WIP/EP	MIC	$\frac{API1}{EDD}$	$\frac{API1}{SLK}$	$\frac{API1}{E/S}$	$\frac{API1}{GI}$	$\frac{API1}{GI+}$	$\frac{API1}{PI1}$
Loose	Small	0.25	0.1	5	0.838	0.862	0.886	1.009	1.027	1.068
				25	0.853	0.887	0.899	0.981	1.015	1.066
			0.25	5	0.908	0.923	0.936	1.01	1.015	1.039
				25	0.909	0.932	0.939	0.970	1.000	1.040
		0.75	0.1	5	0.853	0.864	0.891	0.997	1.014	1.046
				25	0.863	0.881	0.902	0.997	1.013	1.047
			0.25	5	0.920	0.929	0.944	1.00	1.010	1.027
				25	0.922	0.936	0.947	1.00	1.013	1.029
	Wide	0.25	0.1	5	0.894	0.902	0.928	0.950	0.974	1.035
				25	0.910	0.926	0.942	0.90	0.961	1.033
			0.25	5	0.943	0.948	0.961	0.972	0.987	1.017
				25	0.945	0.956	0.964	0.914	0.962	1.018
		0.75	0.1	5	0.912	0.914	0.940	0.961	0.976	1.013
				25	0.920	0.928	0.948	0.959	0.977	1.011
			0.25	5	0.956	0.958	0.971	0.984	0.992	1.008
				25	0.958	0.963	0.974	0.979	0.990	1.009
Tight	Small	0.25	0.1	5	0.737	0.666	0.751	0.922	0.949	1.064
				25	0.740	0.670	0.754	0.874	0.903	1.061
			0.25	5	0.796	0.728	0.805	0.931	0.951	1.045
				25	0.798	0.730	0.807	0.891	0.914	1.043
		0.75	0.1	5	0.794	0.766	0.815	0.966	0.984	1.047
				25	0.798	0.775	0.822	0.955	0.975	1.052
			0.25	5	0.862	0.835	0.876	0.976	0.987	1.030
				25	0.863	0.839	0.878	0.968	0.979	1.031
	Wide	0.25	0.1	5	0.750	0.698	0.761	0.896	0.913	1.032
				25	0.756	0.709	0.769	0.834	0.869	1.030
			0.25	5	0.803	0.757	0.811	0.918	0.932	1.021
				25	0.806	0.763	0.815	0.859	0.889	1.020
		0.75	0.1	5	0.822	0.789	0.836	0.939	0.951	1.017
				25	0.828	0.800	0.844	0.928	0.945	1.017
			0.25	5	0.879	0.854	0.889	0.960	0.968	1.012
				25	0.883	0.860	0.893	0.953	0.964	1.015

Table 3.3(b) Average Performance of heuristics relative to API1 for 20 Jobs:

Due dates	Range	E/T	WIP/EP	MIC	API1/EDD	API1/SLK	API1/E/S	API1/GI	API1/GI+	API1/PI1
Loose	Small	0.25	0.1	5	0.848	0.863	0.876	1.024	1.039	1.097
				25	0.863	0.881	0.892	1.015	1.037	1.088
			0.25	5	0.917	0.926	0.934	1.014	1.021	1.046
				25	0.921	0.932	0.938	1.004	1.018	1.045
		0.75	0.1	5	0.848	0.858	0.872	1.005	1.017	1.060
				25	0.857	0.868	0.881	1.005	1.016	1.060
			0.25	5	0.923	0.929	0.937	1.005	1.010	1.031
				25	0.925	0.933	0.940	1.005	1.011	1.032
	Wide	0.25	0.1	5	0.913	0.916	0.934	0.960	0.970	1.032
				25	0.927	0.933	0.945	0.949	0.974	1.027
			0.25	5	0.959	0.961	0.970	0.984	0.989	1.024
				25	0.961	0.965	0.971	0.967	0.985	1.023
		0.75	0.1	5	0.922	0.925	0.940	0.966	0.976	1.107
				25	0.929	0.933	0.946	0.967	0.977	1.095
			0.25	5	0.965	0.967	0.974	0.985	0.990	1.131
				25	0.967	0.969	0.975	0.984	0.989	1.103
Tight	Small	0.25	0.1	5	0.740	0.690	0.750	0.935	0.956	1.096
				25	0.741	0.694	0.752	0.920	0.943	1.093
			0.25	5	0.815	0.771	0.823	0.949	0.961	1.050
				25	0.815	0.772	0.823	0.930	0.945	1.049
		0.75	0.1	5	0.791	0.776	0.806	0.980	0.991	1.065
				25	0.792	0.780	0.809	0.973	0.985	1.066
			0.25	5	0.873	0.860	0.884	0.987	0.994	1.039
				25	0.874	0.861	0.884	0.985	0.990	1.038
	Wide	0.25	0.1	5	0.739	0.683	0.746	0.901	0.917	1.033
				25	0.750	0.698	0.757	0.886	0.911	1.031
			0.25	5	0.816	0.770	0.821	0.941	0.946	1.055
				25	0.820	0.777	0.825	0.913	0.933	1.054
		0.75	0.1	5	0.830	0.800	0.841	0.943	0.958	1.022
				25	0.837	808	0.847	0.946	0.958	1.095
			0.25	5	0.898	0.874	0.903	0.969	0.974	1.131
				25	0.899	0.877	0.905	0.966	0.973	1.122

Table 3.3(c) Average Performance of heuristics relative to API1 for 30 Jobs:

Due dates	Range	E/T	WIP/EP	MIC	$\frac{API1}{EDD}$	$\frac{API1}{SLK}$	$\frac{API1}{E/S}$	$\frac{API1}{GI}$	$\frac{API1}{GI+}$	$\frac{API1}{PI1}$
Loose	Small	0.25	0.1	5	0.852	0.859	0.870	1.036	1.045	1.106
				25	0.866	0.875	0.884	1.032	1.043	1.098
			0.25	5	0.925	0.929	0.935	1.022	1.026	1.051
				25	0.929	0.934	0.939	1.019	1.026	1.050
		0.75	0.1	5	0.853	0.857	0.866	1.014	1.022	1.067
				25	0.860	0.865	0.873	1.011	1.021	1.065
			0.25	5	0.928	0.930	0.935	1.008	1.013	1.034
				25	0.930	0.933	0.938	1.008	1.012	1.034
	Wide	0.25	0.1	5	0.925	0.934	0.942	0.964	0.972	1.030
				25	0.936	0.945	0.951	0.959	0.974	1.026
			0.25	5	0.967	0.971	0.975	0.986	0.990	1.014
				25	0.969	0.974	0.976	0.978	0.988	1.013
		0.75	0.1	5	0.936	0.941	0.949	0.972	0.981	1.017
				25	0.940	0.946	0.953	0.971	0.981	1.016
			0.25	5	0.972	0.975	0.978	0.987	0.991	1.008
				25	0.973	0.976	0.979	0.986	0.990	1.008
Tight	Small	0.25	0.1	5	0.767	0.740	0.776	0.981	0.993	1.123
				25	0.769	0.743	0.777	0.972	0.986	1.123
			0.25	5	0.849	0.825	0.855	0.978	0.986	1.064
				25	0.851	0.827	0.856	0.970	0.980	1.065
		0.75	0.1	5	0.806	0.802	0.818	1.007	1.018	1.085
				25	0.808	0.805	0.820	1.003	1.014	1.086
			0.25	5	0.890	0.885	0.897	1.003	1.008	1.045
				25	0.891	0.886	0.898	1.000	1.005	1.046
	Wide	0.25	0.1	5	0.748	0.703	0.753	0.906	0.918	1.040
				25	0.760	0.718	0.765	0.897	0.917	1.038
			0.25	5	0.836	0.801	0.839	0.942	0.950	1.023
				25	0.840	0.807	0.843	0.930	0.945	1.023
		0.75	0.1	5	0.847	0.821	0.853	0.942	0.953	1.023
				25	0.852	0.823	0.858	0.941	0.953	1.023
			0.25	5	0.912	0.895	0.915	0.969	0.974	1.012
				25	0.914	0.897	0.917	0.967	0.973	1.012

Table 3.3(d) Average Performance of heuristics relative to API1 for 40 Jobs:

Due dates	Range	E/T	WIP/EP	MIC	$\frac{API1}{EDD}$	$\frac{API1}{SLK}$	$\frac{API1}{E/S}$	$\frac{API1}{GI}$	$\frac{API1}{GI+}$	$\frac{API1}{PI1}$
Loose	Small	0.25	0.1	5	0.847	0.850	0.858	1.040	1.048	1.113
				25	0.860	0.864	0.871	1.037	1.047	1.107
			0.25	5	0.923	0.925	0.930	1.025	1.028	1.055
				25	0.927	0.930	0.934	1.024	1.029	1.054
		0.75	0.1	5	0.847	0.848	0.856	1.020	1.028	1.075
				25	0.853	0.855	0.862	1.019	1.027	1.075
			0.25	5	0.926	0.927	0.931	1.012	1.015	1.038
				25	0.929	0.930	0.934	1.012	1.015	1.039
	Wide	0.25	0.1	5	0.933	0.935	0.945	0.965	0.971	1.033
				25	0.941	0.944	0.952	0.961	0.972	1.028
			0.25	5	0.972	0.972	0.977	0.987	0.990	1.015
				25	0.973	0.974	0.978	0.983	0.989	1.014
		0.75	0.1	5	0.942	0.943	0.950	0.975	0.982	1.020
				25	0.945	0.947	0.954	0.974	0.982	1.019
			0.25	5	0.976	0.976	0.980	0.989	0.991	1.010
				25	0.976	0.977	0.980	0.988	0.991	1.009
Tight	Small	0.25	0.1	5	0.765	0.746	0.772	0.993	1.001	1.137
				25	0.767	0.749	0.774	0.985	0.996	1.137
			0.25	5	0.854	0.837	0.858	0.990	0.995	1.069
				25	0.855	0.839	0.859	0.982	0.988	1.071
		0.75	0.1	5	0.798	0.794	0.806	1.018	1.026	1.093
				25	0.799	0.797	0.808	1.016	1.024	1.096
			0.25	5	0.889	0.885	0.894	1.009	1 013	1.050
				25	0.890	0.887	0.895	1.009	1.012	1.052
	Wide	0.25	0.1	5	0.765	0.730	0.769	0.915	0.924	1.042
				25	0.776	0.743	0.780	0.909	0.925	1.040
			0.25	5	0.856	0.831	0.858	0.952	0.958	1.022
				25	0.860	0.835	0.862	0.947	0.958	1.021
		0.75	0.1	5	0.859	0.842	0.865	0.951	0.960	1.025
				25	0.864	0.847	0.869	0.950	0.959	1.025
			0.25	5	0.924	0.912	0.926	0.974	0.978	1.012
				25	0.925	0.914	0.927	0.972	0.977	1.012

Table 3.3(e) Average Performance of heuristics relative to API1 for 50 Jobs:

Due dates	Range	E/T	WIP/EP	MIC	$\frac{API1}{EDD}$	$\frac{API1}{SLK}$	$\frac{API1}{E/S}$	$\frac{API1}{GI}$	$\frac{API1}{GI+}$	$\frac{API1}{PI}$
Loose	Small	0.25	0.1	5	1.237	1.233	1.220	1.048	1.007	0.942
				25	1.217	1.211	1.200	1.047	1.009	0.945
			0.25	5	1.113	1.111	1.1.5	1.028	1.003	0.975
				25	1.110	1.106	1.101	1.029	1.005	0.976
		0.75	0.1	5	1.213	1.212	1.201	1.028	1.008	0.956
				25	1.204	1.201	1.191	1.027	1.008	0.955
			0.25	5	1.095	1.095	1.090	1.015	1.003	0.977
				25	1.093	1.092	1.087	1.015	1.003	0.977
	Wide	0.25	0.1	5	1.040	1.038	1.027	0.971	1.006	0.940
				25	1.033	1.030	1.021	0.972	1.012	0.945
			0.25	5	1.018	1.017	1.013	0.990	1.002	0.975
				25	1.017	1.015	1.012	0.989	1.007	0.976
		0.75	0.1	5	1.043	1.042	1.034	0.982	1.008	0.963
				25	1.039	1.037	1.029	0.982	1.008	0.963
			0.25	5	1.016	1.015	1.012	0.991	1.003	0.982
				25	1.015	1.014	1.011	0.991	1.003	0.982
Tight	Small	0.25	0.1	5	1.308	1.341	1.297	1.001	1.008	0.880
				25	1.298	1.330	1.287	0.996	1.011	0.876
			0.25	5	1.165	1.188	1.160	0.995	1.005	0.931
				25	1.156	1.178	1.151	0.988	1.006	0.923
		0.75	0.1	5	1.286	1.292	1.274	1.026	1.008	0.939
				25	1.281	1.286	1.268	1.024	1.008	0.935
			0.25	5	1.140	1.144	1.134	1.013	1.004	0.964
				25	1.137	1.142	1.131	1.012	1.004	0.962
	Wide	0.25	0.1	5	1.208	1.265	1.201	0.924	1.010	0.887
				25	1.192	1.245	1.186	0.925	1.019	0.890
			0.25	5	1.119	1.153	1.116	0.958	1.006	0.938
				25	1.114	1.146	1.111	0.958	1.012	0.938
		0.75	0.1	5	1.118	1.141	1.110	0.960	1.010	0.937
				25	1.111	1.132	1.104	0.959	1.010	0.936
			0.25	5	1.058	1.072	1.055	0.978	1.004	0.966
				25	1.057	1.069	1.053	0.977	1.005	0.965

Table 3.4(a) Effect of Tardiness Factor: Performance of heuristics relative to |API1| averaged over all combinations of the other factors

Number of Jobs	Tardiness Factor	API EDD	API SLK	API E/S	API GI	API GI+	API PI
10	Low	0.906	0.919	0.936	0.974	0.995	1.032
	High	0.807	0.765	0.820	0.923	0.942	1.034
20	Low	0.915	0.923	0.933	0.990	1.001	1.063
	High	0.815	0.781	0.8234	0.945	0.958	1.071
30	Low	0.923	0.928	0.934	0.997	1.005	1.040
	High	0.834	0.812	0.840	0.963	0.973	1.052
40	Low	0.923	0.925	0.931	1.001	1.007	1.044
	High	0.840	0.824	0.845	0.973	0.981	1.057
50	Low	0.927	0.929	0.933	1.007	1.012	1.048
	High	0.850	0.837	0.853	0.985	0.992	1.061

Table 3.4(b) Effect of due date range: Performance of heuristics relative to |API1| averaged over all combinations of the other factors

Number of Jobs	Due date Range	API EDD	API SLK	API E/S	API GI	API GI+	API PI
10	Low	0.841	0.827	0.866	0.966	0.984	1.046
	High	0.873	0.858	0.8904	0.932	0.953	1.020
20	Low	0.847	0.837	0.863	0.984	0.996	1.060
	High	0.883	0.866	0.894	0.952	0.964	1.074
30	Low	0.861	0.856	0.871	1.004	1.012	1.072
	High	0.895	0.883	0.903	0.956	0.966	1.020
40	Low	0.858	0.854	0.865	1.012	1.018	1.079
	High	0.905	0.895	0.911	0.962	0.969	1.022
50	Low	0.863	0.862	0.870	1.028	1.033	1.090
	High	0.913	0.904	0.916	0.965	0.972	1.020

Table 3.4(c) Effect of EP/TP: Performance of heuristics relative to [API1] averaged over all combinations of the other factors

Number of Jobs	EP/TP	API LDD	API SLK	API E/S	API GI	API GI+	API PI
10	Low	0.837	0.816	0.858	0.927	0.954	1.040
	High	0.877	0.868	0.898	0.971	0.984	1.026
20	Low	0.847	0.827	0.860	0.956	0.972	1.053
	High	0.883	0.876	0.897	0.979	0.988	1.081
30	Low	0.862	0.849	0.871	0.973	0.984	1.056
	High	0.894	0.890	0.903	0.987	0.994	1.036
40	Low	0.867	0.857	0.874	0.981	0.989	1.060
	High	0.896	0.893	0.902	0.993	0.999	1.041
50	Low	0.875	0.867	0.880	0.991	0.998	1.065
	High	0.901	0.899	0.906	1.002	1.007	1.045

Table 3.4(d) Effect of WIP/EP: Performance of heuristics relative to [API1] averaged over all combinations of the other factors

Number of Jobs	WIP/EP	API LDD	API SLK	API E/S	API GI	API GI+	API PI
10	Low	0.829	0.815	0.856	0.942	0.965	1.040
	High	0.884	0.870	0.901	0.956	0.972	1.026
20	Low	0.833	0.819	0.850	0.961	0.977	1.073
	High	0.897	0.884	0.907	0.974	0.983	1.061
30	Low	0.845	0.837	0.857	0.976	0.987	1.061
	High	0.911	0.903	0.917	0.985	0.991	1.031
40	Low	0.848	0.840	0.856	0.983	0.992	1.067
	High	0.916	0.910	0.920	0.991	0.996	1.034
50	Low	0.855	0.849	0.861	0.996	1.004	1.074
	High	0.921	0.917	0.925	0.997	1.001	1.036

Table 3.4(e) Effect of Machine idle cost: Performance of heuristics relative to [API1] averaged over all combinations of the other factors

Number of Job	Machine idle cost	API TDD	API SLK	API [illegible] S	API GI	API GI+	API PI
10	Low	0.856	0.842	0.878	0.949	0.969	1.034
	High	0.858	0.842	0.878	0.947	0.967	1.032
20	Low	0.864	0.851	0.877	0.968	0.980	1.065
	High	0.866	0.851	0.878	0.966	0.978	1.066
30	Low	0.877	0.869	0.886	0.981	0.990	1.047
	High	0.879	0.870	0.888	0.978	0.987	1.044
40	Low	0.880	0.873	0.887	0.987	0.994	1.052
	High	0.883	0.875	0.889	0.985	0.992	1.048
50	Low	0.887	0.882	0.892	0.997	1.003	1.056
	High	0.889	0.884	0.894	0.994	1.000	1.052

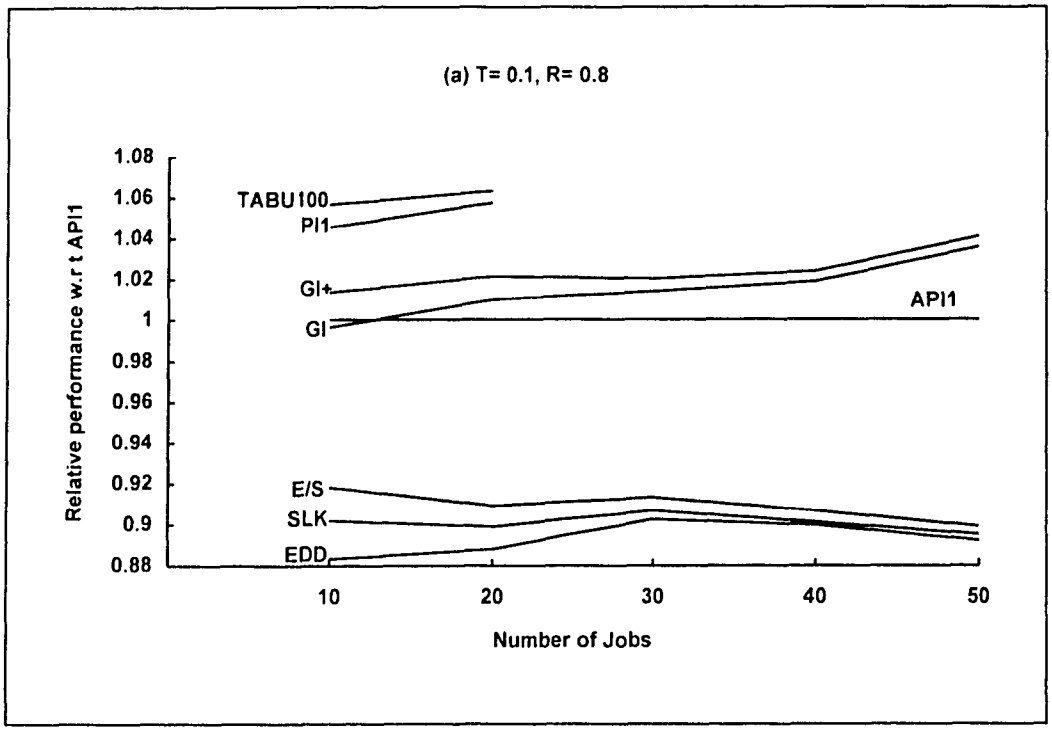


Figure 3.1(a) Loose due dates & small range

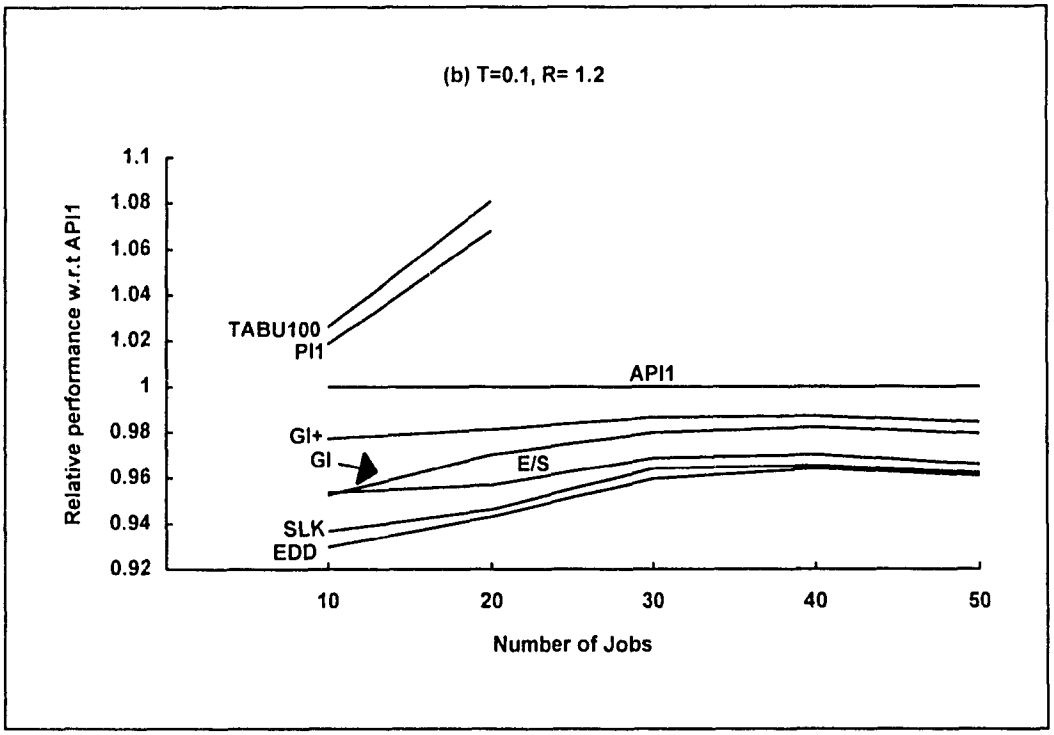


Figure 3.1(b) Loose due dates & wide range

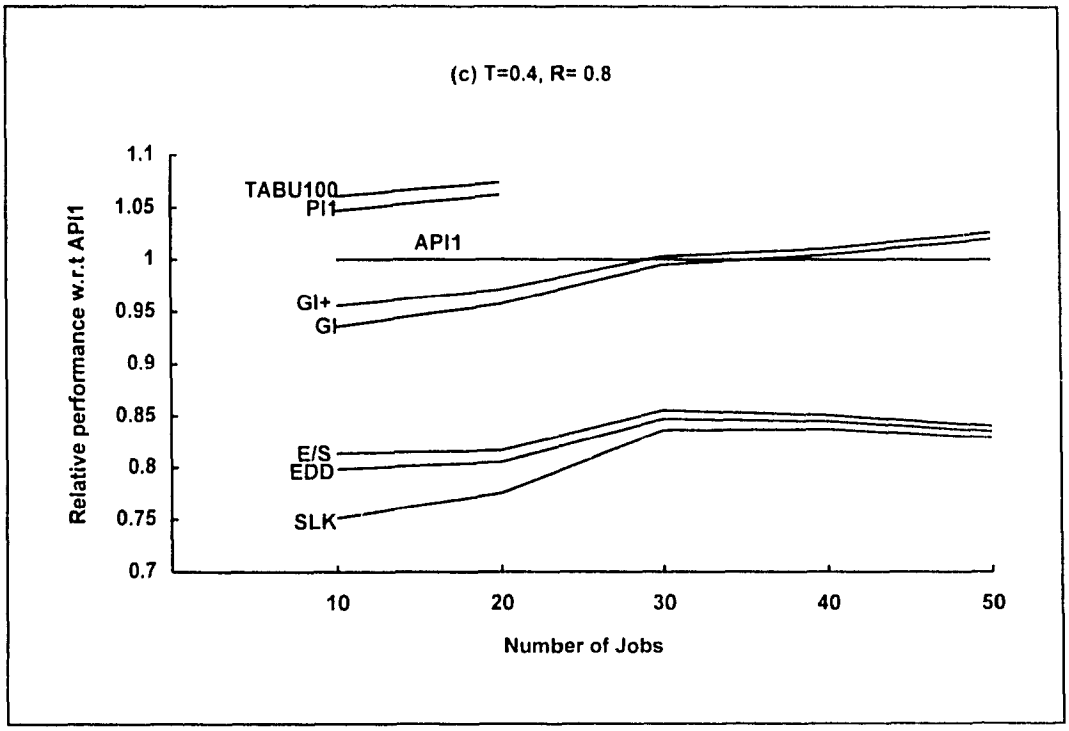


Figure 3.1(c) Tight due dates & small range

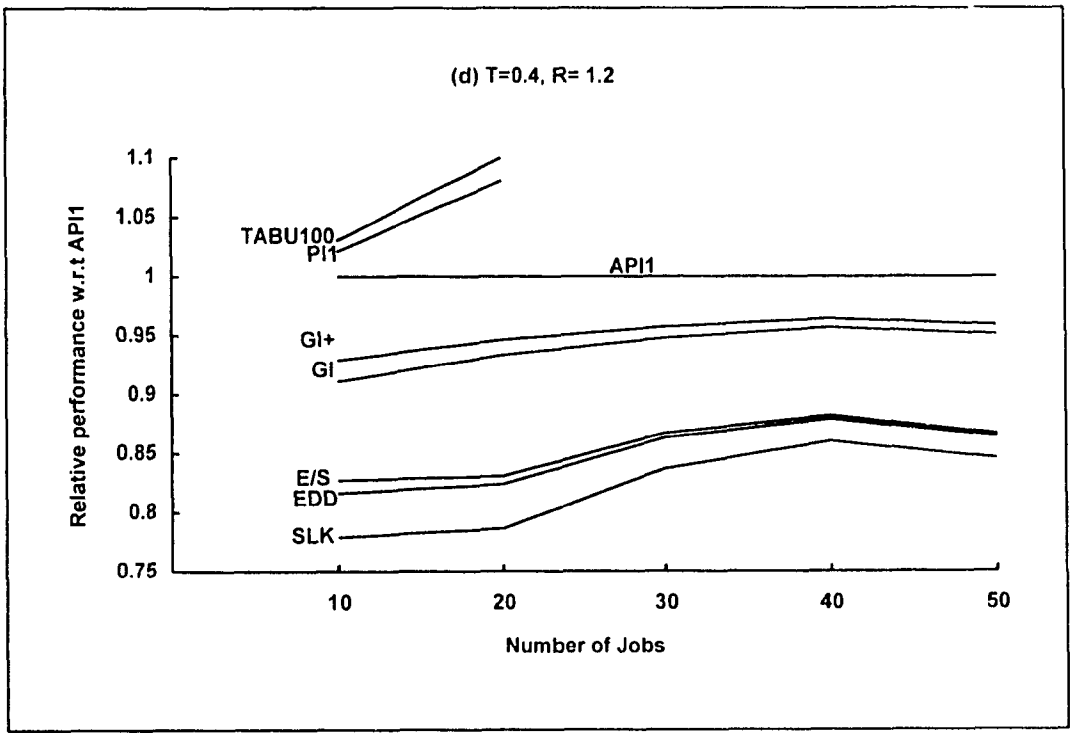


Figure 3.1(d) Tight due dates & wide range

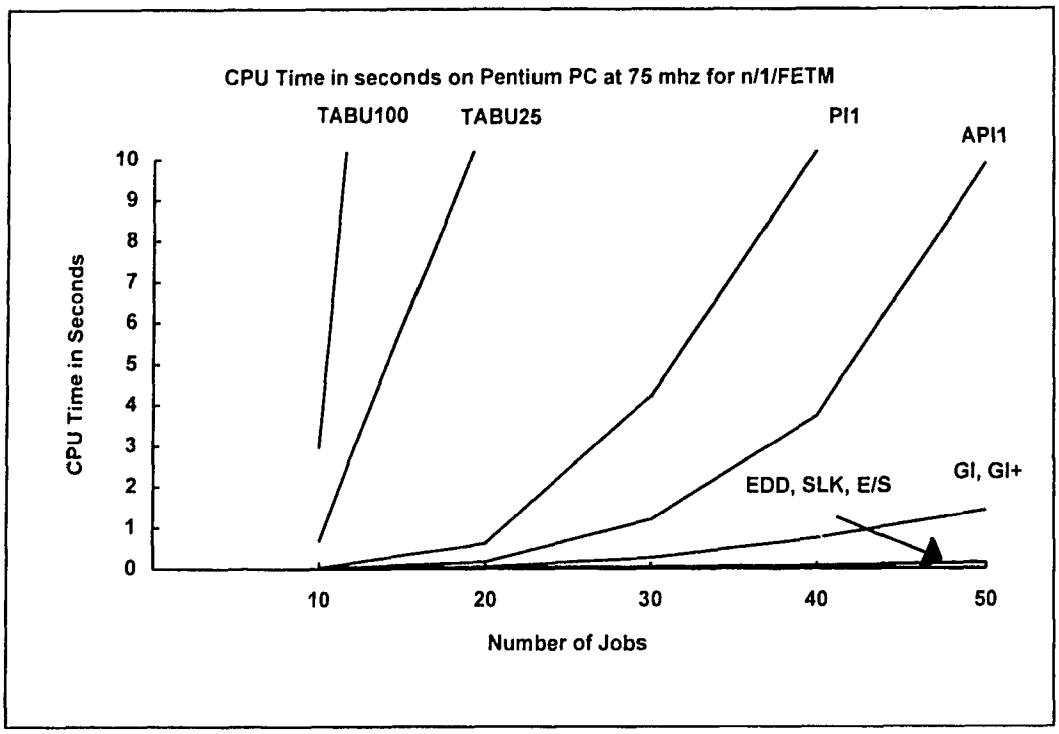


Figure 3.2 Average running time of algorithms

3.8 Conclusion

In this chapter, we developed dominance conditions for scheduling of jobs on a single machine where the objective is to minimize the sum of earliness, tardiness, work-in-process and machine idle costs and used them to develop the GREEDY INSERT heuristic. The computational study showed that these complex combinatorial problems can be solved within reasonable time and reasonable accuracy by using GREEDY INSERT. The results also showed that the relative magnitude of earliness, tardiness, work-in-process penalties did not have significant effect on the relative performance of heuristics when $\beta_i > \alpha_i > \gamma_i$. So, we will eliminate these three factors for future consideration when we generalize the problem to non zero release times and multiple due dates. Although dominance conditions suggest to sort all jobs in non-increasing order of $(\beta_i + \gamma_i)/p_i$ for [GI], we found that the performance of β_i / p_i is better, so all results, unless otherwise specified, are presented for an initial sorting of jobs in non-

increasing order of β_i / p_i. Because of the poor performance, [SPT] will not be considered for remaining work. Since the results showed that there is no significant difference in the performance between heuristics [API1] and [API2] or between heuristics [PI1] and [PI2] we decided to keep only [API2] and [PI2] for future consideration. While search methods like Tabu, [PI2] and [API2] perform better for small problem sizes but they become prohibitive for large problems. Therefore we recommend GREEDY INSERT for large problem sizes. We strongly believe that GREEDY INSERT will provide perfect trade off between solution quality and computational cost for big problem sizes.

CHAPTER 4

Single due date per job and different release times

In this chapter, we extend the single machine scheduling problem of chapter 3 to non zero release times but keep each job with a single due date. We take this intermediate step because this problem has its own intrinsic value as attested by others who study non-zero release times. This problem may help researchers solving more complex job shop scheduling problems for minimizing total cost. So, we first develop an optimal idle time insertion heuristic for the problem, and then we modify [GI] to account for non-zero release times. Based on the results of the previous chapter, we test the modified [GI] heuristic against others.

4.1 Problem Formulation

Consider a non-preemptive, single-machine scheduling problem with n jobs. All jobs with deterministic processing times p_i have possibly different due dates d_i and have possibly different release dates $r_i \geq 0$. Set up times are included in processing times. Idle time is allowed between any successive job operations. Here the objective is to minimize the total cost by minimizing the sum of earliness, tardiness, work in process, and machine idle costs.

Total Cost = Work-In-Process Cost + Earliness Penalty Cost +
Tardiness Penalty Cost + Machine Idle Cost.

We represent single machine scheduling problem with non-zero release times with the objective of minimizing the total of work-in-process, earliness, tardiness, and machine idle costs as $n/1/r_i/FETM$. The function we seek to minimize is the sum of all those costs, and can be expressed as

$$= \sum_{i=1}^{n} (\gamma_i W_i + \alpha_i E_i + \beta_i T_i) + \mu M_i$$

The formulation is identical to that of section 3.1, except for the release times. Unlike the section 3.1, work-in-process can be calculated as $(C_i - r_i)$.

4.2 An Optimal Idle Time Insertion Algorithm

We designed an algorithm to insert idle times optimally in between job operations given their sequence, for $n/1/r_i/FETM$ problem. Our procedure starts with all jobs scheduled, in the given sequence 1, 2,..., n, starting at the release time of the first job and with no inserted idle time. First select the first job u such that difference between completion time of job u and the staring time of job $u+1$ is greater than zero. Starting with the first job, we insert idle time before a job i only if moving jobs i, $i+1$,... , u forward by one time unit results in a reduction in total cost. The amount of idle time we insert is the minimum of two quantities: 1).The smallest non-zero earliness among i, $i+1$,..., u. 2). The difference between the completion time of job u and the start time job $u+1$, since the cost reduction will not change over that range when all costs are linear. Then we calculate the potential cost reduction to see whether inserting more idle time is beneficial. The above procedure is formally stated as follows:

Algorithm IDLE TIME INSERT RELEASE [ITIR]: Schedules n jobs with possibly different release times in a given sequence to minimize the total of flow time, earliness, tardiness, and machine idle costs.

1. Number the jobs 1,..., n in the order of the given sequence, form a schedule with no inserted idle times, and compute their earliness and tardiness.
   (a) Set $C_1 = r_1 + p_1$, $E_1 = \max\{0, d_1 - C_1\}$, and $T_1 = \max\{0, C_1 - d_1\}$
   (b) Set $C_i = \max\{r_i, C_{i-1}\} + p_i$, $E_i = \max\{0, d_i - C_i\}$, and $T_i = \max\{0, C_i - d_i\}$ for $i = 2, \ldots, n$.

2. Initialize three counters $j \leftarrow 1$, $k \leftarrow 1$, $u \leftarrow 1$ and a number to keep track of potential cost reduction from inserted idle time $CR \leftarrow 0$.

3. If $u \neq n$ and there exist a job u from k to n such that there is an idle time between processing of u and $u+1$ compute $diff \leftarrow$ (Starting time of job u+1 - Completion time of job u, $u < n$). Otherwise $u \leftarrow n$.

4. Compute the saving resulting from pushing job k forward one time unit, and add that cost to CR:

$$CR \leftarrow CR + \alpha_k - \gamma_k \qquad \text{if } E_k > 0$$

$$CR \leftarrow CR - \beta_k - \gamma_k \qquad \text{otherwise}$$

5. Check the values of CR and k:

   (a) If $CR > 0$ and $k < u$, set $k \leftarrow k+1$ and go to step 4.

   (b) If $CR \leq 0$ and $k < u$, set $j \leftarrow k+1$, $k \leftarrow k+1$, $CR \leftarrow 0$ and go to step 4.

   (c) If $CR > \mu$ and $u = n$, move all the jobs $j,\ldots, n$ forward in the time by $\min_{i=j,\ldots,n,\ E_i>0} E_i$. Update the earliness and tardiness of the moved jobs, set $k \leftarrow j$, $CR \leftarrow 0$ and go to step 4.

   (d) If $u \neq n$, move all the jobs $j,\ldots, u$ forward in the time by $\min\{diff, \min_{i=1,\ldots,u,\ E_i>0} E_i\}$. Update the earliness and tardiness of the moved jobs, set $k \leftarrow j$, $CR \leftarrow 0$ and go to step 3.

   (e) If $CR \leq \mu$ and $k = n$, stop.

In step 5(b), since the cost increases by moving jobs $j,\ldots, k$ forward, we stop considering those jobs for idle time insertion and continue starting with job k+1. Note that each time step 5(c) or 5(d) applies, it is not necessary that a job which was early becomes on-time, because we are taking the *diff* into consideration. At most we need ($2n$-1) iterations to make all jobs on time or tardy. In each iteration, there will be at most n jobs to consider, so the time complexity of the optimal timing algorithm is O (n^2).

Theorem 2 *For a given sequence of jobs, algorithm "IDLE TIME INSERT RELEASE" inserts idle times optimally in between jobs when all costs are linear.*

Proof: Same as the proof of theorem 1. ❑

4.3 A Job Insertion Heuristic

Here we adopt GREEDY INSERT procedure for the $n/1/r_i/FETM$ problem. The following principle should be honored throughout the [GI] procedure to accommodate non-zero release times:

Starting time of any job should be at least greater than or equal to the release time of that job.

Otherwise, all the details are the same as before.

4.4 Experimental Design

In order to test the performance of our heuristic and compare it to other procedures, we designed a 2^2 factorial experiment which takes T & R into account because of their significant effect on the performance of heuristics. In order to be consistent with results reported elsewhere, we randomly generated the processing times of the jobs from a Discrete Uniform [1, 30] distribution, the release times from a Discrete Uniform [0, $P/2$] distribution where $P = \sum_{i=1}^{n} p_i$, the due dates by adding r_i to a Discrete Uniform [$P(1- T - R/2)$, $P(1 - T + R/2)$] distribution, the tardiness costs per unit time β_i from a Continuous Uniform[0.5, 5]P/n distribution, the earliness cost per unit time α_i form a Continuous Uniform [0.25, 0.75]β_i distribution, the work-in-process cost per unit time γ_i from a Continuous Uniform [0.1, 0.25]α_i distribution and the machine idle cost is generated from a Discrete Uniform [5, 25] distribution. We ran the experiment for instances with 10, 20, 30, 40, 50, and 75 jobs per problem and tested GREEDY INSERT against some of the heuristics that have been introduced in the last chapter for solving earliness, tardiness flowtime, and machine idle cost problems with different due dates,

and that can be easily modified to accommodate with non-zero release times. In a first phase, we ran a few cases to determine the number of runs required for the average performance to converge. We settled on 200 runs for each case which gave us a confidence interval of $\pm 2-3\%$ around the average performance. Since we are conducting four experiments for two factors we generated a set of 800 problems for each instance. All the algorithms presented in this chapter were coded in PASCAL and run on a Pentium at 66 MHz.

4.5 Experimental Results

The exact average total cost figures obtained from running 200 problems for all heuristics at different T and R values are tabulated in appendix-B which includes tables B.1, B.2, B.3, B.4. Average CPU times are tabulated in B.5. The average relative performance of heuristics with respect to [API1] is presented in Table 4.1. The average relative performance of heuristics with respective [API2] for different cases is plotted in the Figures 4.1(a), 4.1(b), 4.1(c), and 4.1(d).

Some of the interesting results of our study are mentioned below:

- The performance of GREEDY INSERT for small problem sizes seems to be a little inferior to that of [API2]. But, as the problem size gets larger, performance becomes superior to that of [API2], for small R, and very close for wide ranges. This performance can be noticed from Figures 4.1(a), 4.1(b), 4.1(c), 4.1(d).

- CPU time required for GREEDY INSERT increases almost linearly as problem size increases. But for [API2], [PI2], and Tabu Search, CPU time seems to grow exponentially with the problem size. This can be clearly seen from Figure 4.2.

- Performance of GREEDY INSERT is superior to [EDD], [SLK], [F/S] for all tardiness and range factors.

- While [EDD] performs slightly better than that of [SLK] for tight due dates (T 0.4) at wide ranges (R=1.2), [SLK] seems to do well for loose due dates (T 0.1) at wide ranges (R=1.2).

- GREEDY INSERT runs much faster than search heuristics. For instance GREEDY INSERT solved 75 jobs problem as much as 150 times faster than [API2] and performed superior to [API2] for small ranges (R=0.8), close to [API2] for wide ranges (R 1.2).

- Performance of Tabu Search is greater than all other heuristics presented, but it is very expensive in terms of computation.

Table 4.1 Average performance of heuristics relative to API2:

No jobs	Due dates	Range	API2/EDD	API2/SLK	API2/ES	API2/GI	API2/GI+	API2/PI2
10	Loose	Small	0.893	0.899	0.917	0.974	0.984	1.008
		Wide	0.929	0.936	0.950	0.963	0.981	1.007
	Tight	Small	0.813	0.799	0.854	0.930	0.947	1.020
		Wide	0.805	0.761	0.831	0.952	0.961	1.034
20	Loose	Small	0.903	0.900	0.916	0.987	0.994	1.018
		Wide	0.952	0.956	0.964	0.981	0.990	1.007
	Tight	Small	0.865	0.846	0.879	0.954	0.966	1.025
		Wide	0.851	0.824	0.873	0.968	0.980	1.029
30	Loose	Small	0.905	0.907	0.914	0.995	1.002	1.022
		Wide	0.960	0.960	0.967	0.986	0.993	1.006
	Tight	Small	0.870	0.871	0.882	0.975	0.986	1.031
		Wide	0.903	0.896	0.915	0.971	0.981	1.016
40	Loose	Small	0.899	0.901	0.907	1.000	1.007	1.028
		Wide	0.960	0.961	0.965	0.986	0.992	1.007
	Tight	Small	0.864	0.867	0.873	0.983	0.994	1.040
		Wide	0.877	0.867	0.889	0.970	0.980	1.018
50	Loose	Small	0.898	0.900	0.904	1.007	1.012	1.030
		Wide	0.965	0.966	0.969	0.990	0.995	1.007
	Tight	Small	0.861	0.863	0.868	0.991	0.999	1.043
		Wide	0.911	0.908	0.922	0.981	0.990	1.015
75	Loose	Small	0.905	0.906	0.908	1.014	1.017	-----
		Wide	0.973	0.973	0.976	0.994	0.997	-----
	Tight	Small	0.871	0.871	0.875	1.003	1.009	-----
		Wide	0.931	0.925	0.936	0.984	0.991	-----

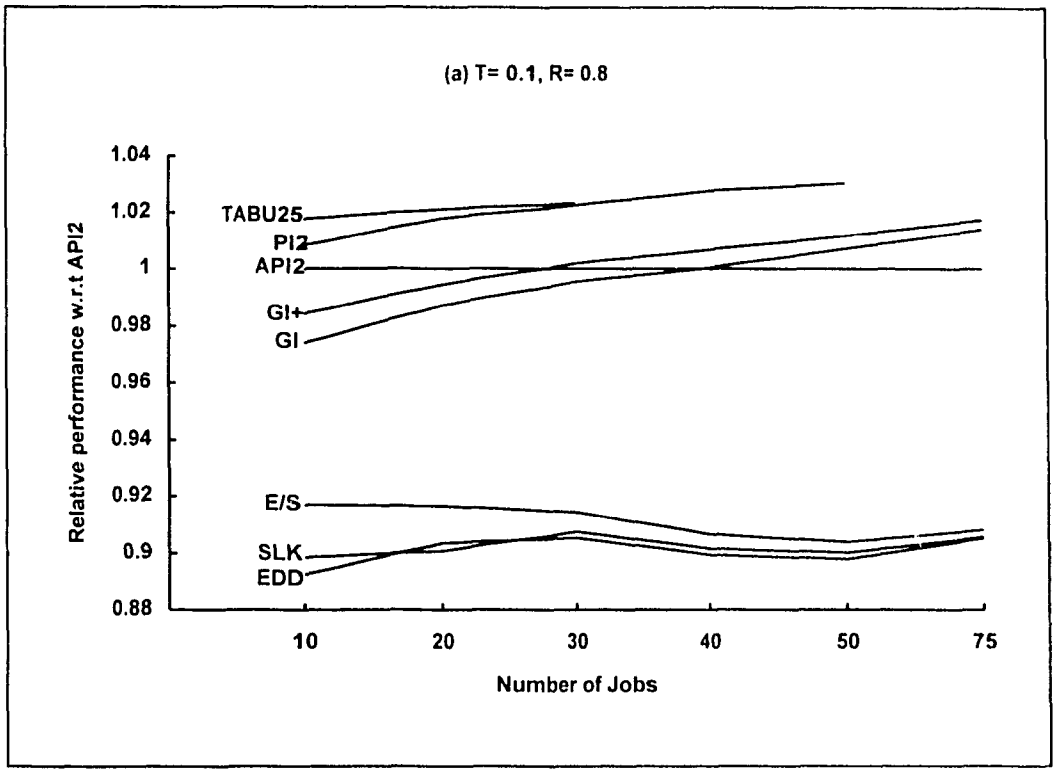


Figure 4.1(a) Loose due dates & small range

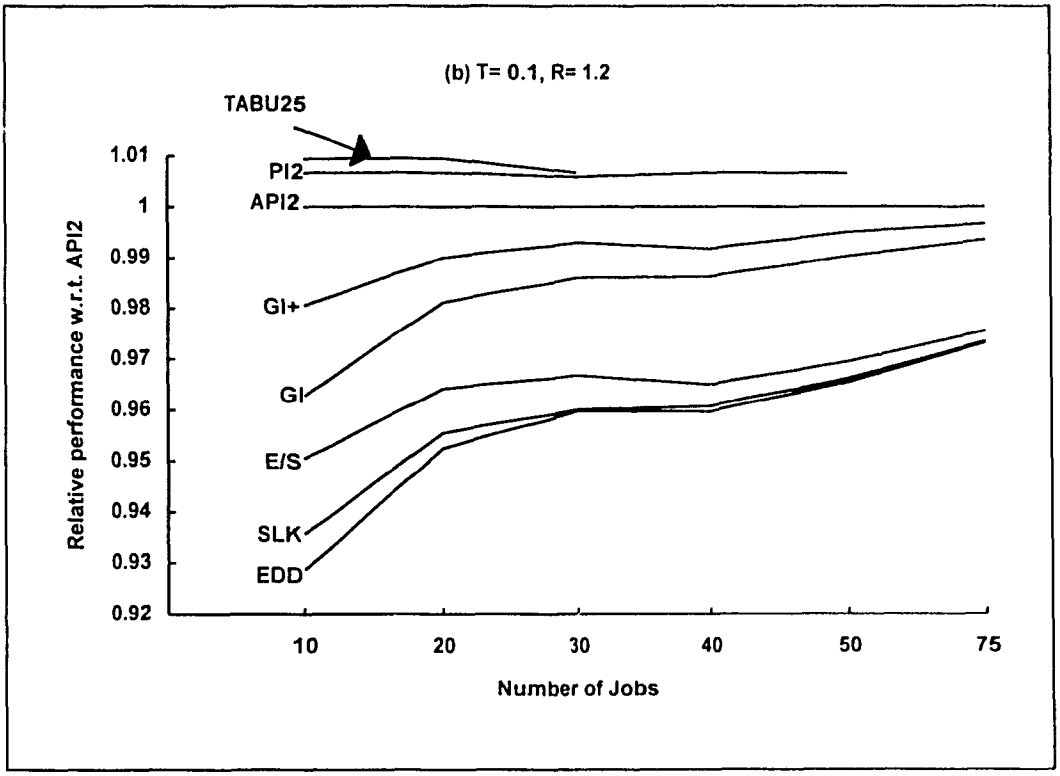


Figure 4.1(b) Loose due dates & wide range

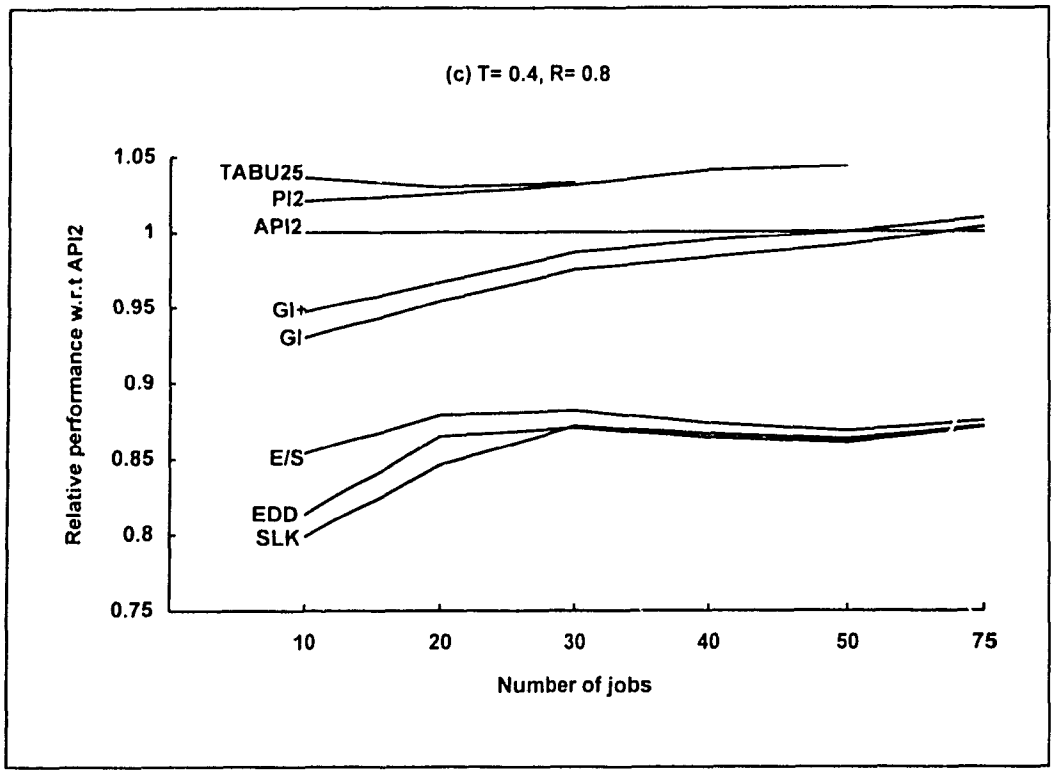


Figure 4.1(c) Tight due dates & small range

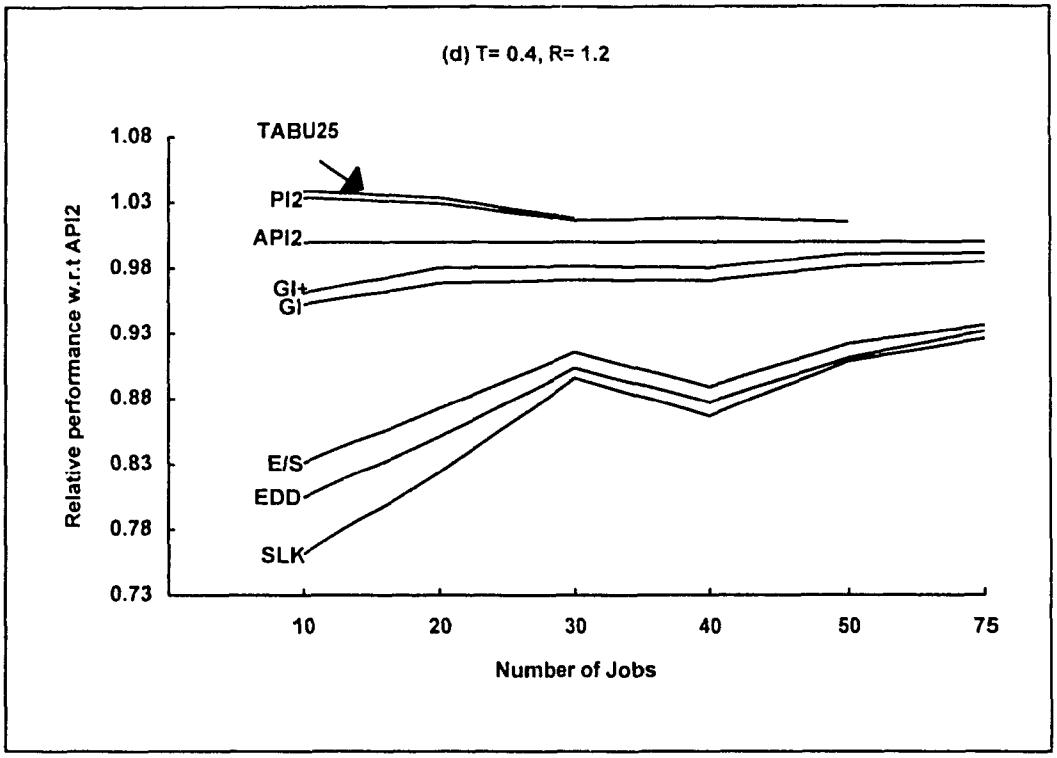


Figure 4.1(d) Tight due dates & wide range

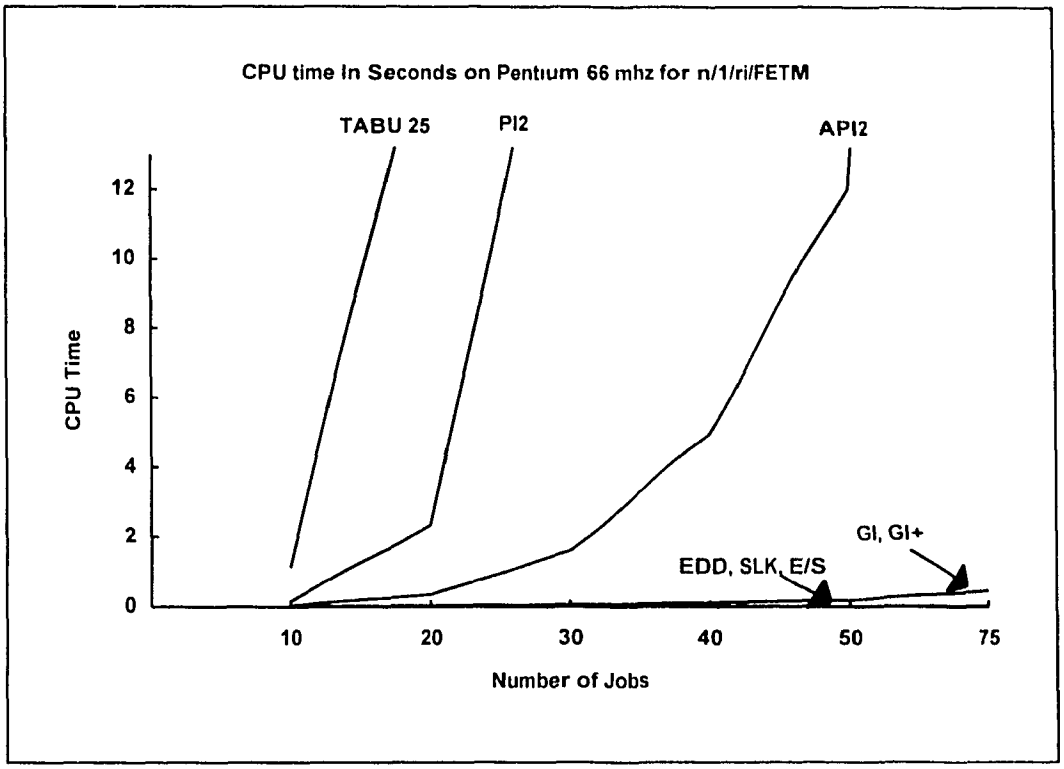


Figure 4.2 Average running time of algorithms

4.6 Conclusion

In this chapter, we modified & tested the GREEDY INSERT heuristic against other procedures that we could adopt for solving the single machine scheduling problem to minimize work-in-process, earliness, tardiness, and machine idle costs where release times of jobs are greater than or equal to zero under different experimental conditions. We also provided a new heuristic procedure to insert idle times optimally for a given sequence on a single machine with non-zero release times. The computational results showed that GREEDY INSERT will be an alternative to solve large problem sizes efficiently with low computational cost. The efficiency of the results obtained for this problem by GREEDY INSERT strengthen the idea of extending this heuristic procedure to the decomposed job shop cost problem, which will be examined in the next chapter.

CHAPTER 5

Multiple due dates per job and different release times

In this chapter, we extend our previous single machine scheduling problem to account for the case where each job can have multiple due dates. This is the problem that was obtained by Amiouny (1995) when decomposing the job shop scheduling problem with the objective of minimizing total costs into single machine problems in a manner similar to the Shifting Bottleneck procedure of Adams, Balas and Zawack (1988). For consistency with earlier chapters, we first formulate a model, then develop an optimal idle time insertion heuristic for a given sequence, and then design new heuristic procedures for the problem. Finally, we test the new heuristic procedures against others.

5.1 Problem Formulation

Consider a non-preemptive, single machine problem with n jobs. All jobs or operations with deterministic processing times p_i have possibly different multiple due dates d_{im} where m is the maximum number of due dates of any job, which is different for different jobs and have possibly different release times $r_i \geq 0$. Each job will have multiple tardiness penalties β_{im} corresponding to multiple due dates. We assume without loss of generality that the multiple due dates for job i are numbered in such a way that $d_{i1} < d_{i2} < \ldots < d_{im}$ and the corresponding tardiness penalties are such that $\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}$. Set up times are included in processing times. Idle time is allowed between any successive job operations. Here the objective is to minimize the total cost by minimizing the sum of the work-in-process, earliness, tardiness, and machine idle costs.

Total Cost = Work-In-Process Cost + Earliness Penalty Cost +
Tardiness Penalty Cost + Machine Idle Cost.

In the case of Amiouny's reduction, $m \leq n$ and $\beta_{i1} < \beta_{i2} < \ldots < \beta_{im}$. Also that Amiouny suggests an artificial machine idle cost to avoid slack schedules at the beginning of the solution procedure.

We represent the single machine scheduling problem with possibly non-zero release times and multiple due dates, and with the objective of minimizing the total of work-in-process, earliness, tardiness, and machine idle costs as $n/1/r_i / d_{im} / PETM$. The function we seek to minimize is the sum of all those costs, and can be expressed as

$$= \sum_{i=1}^{n} (\gamma_i W_i + \alpha_i E_i + [(C_i - d_{ik})\beta_{ik} + (d_{ik} - d_{ik-1})\beta_{ik-1} + \ldots + (d_{i2} - d_{i1})\beta_{i1}]) + \mu M_i$$

where $d_{ik+1} \geq C_i > d_{ik}$

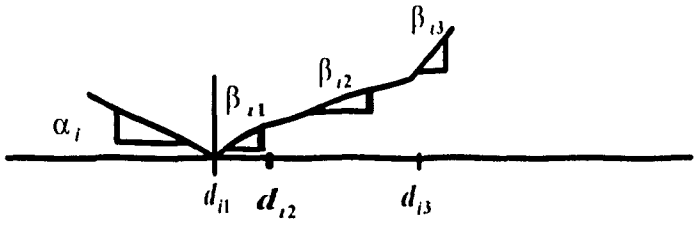


Figure 5.1 Representation of penalties for decomposed job shop cost problem

All the costs are exactly calculated like the section 4.1 except tardiness cost. Here the tardiness cost is a piecewise linear function where the number of terms depends on the completion time of the job. A penalty is incurred only for due dates that fall before completion time and after its first due date. The total tardiness cost of a given job is summation of all independent tardiness costs which takes place between any of two consequent due dates and those due dates less than or equal to completion time of the job.

5.2 An Optimal Idle Time Insertion Algorithm

We designed an algorithm to insert idle times optimally in between job operations for a given sequence. This algorithm is similar to the one presented in the last chapter. The main difference is that tardiness costs are not linear as before. So,

when we insert idle time, we also take the difference between completion time to nearest due date into consideration. The details are as follows:

Algorithm IDLE TIME INSERT MUL DUE DATES [ITIMD]: Schedules n jobs with possibly different release times and multiple due dates in a given sequence to minimize the total of flow time, earliness, tardiness, and machine idle costs.

1. Number the jobs 1,..., n in the order of the given sequence, form a schedule with no inserted idle times, and compute their earliness and tardiness.
   (a) Set $C_1 = r_1 + p_1$, $E_1 = \max\{0, d_{11} - C_1\}$, and $T_1 = \max\{0, C_1 - d_{11}\}$
   (b) Set $C_i = \max\{r_i, C_{i-1}\} + p_i$, $E_i = \max\{0, d_{i1} - C_i\}$, and $T_i = \max\{0, C_i - d_{i1}\}$ for $i = 2,\ldots,n$.
2. Initialize three counters $j \leftarrow 1$, $k \leftarrow 1$, $u \leftarrow 1$ and a number to keep track of potential cost reduction from inserted idle time $CR \leftarrow 0$.
3. If there exists a job u ($k \le u < n$) such that there is an idle time between processing of u and $u+1$ compute $diff \leftarrow$ (Starting time of job $(u+1)$ − Completion time of job u, $u < n$). Otherwise set $u \leftarrow n$.
4. Compute the saving resulting from pushing job k forward one time unit, and add that cost to CR:

$$CR \leftarrow CR + \alpha_k - \gamma_k \qquad \text{if } E_k > 0$$

$$CR \leftarrow CR - \beta_{km} - \gamma_k \qquad \text{otherwise}$$

5. Check the values of CR, k and u:
   (a) If $CR > 0$ and $k < u$, set $k \leftarrow k+1$ and go to step 4.
   (b) If $CR \le 0$ and $k < u$, set $j \leftarrow k+1$, $k \leftarrow k+1$, $CR \leftarrow 0$ and go to step 4.
   (c) If $CR > \mu$ and $u = n$, calculate $gap \leftarrow \min_{\substack{j \le i \le u \\ d_{im} > C_i}} \{d_{im} - C_i\}$ (where d_{im} is the first due date of job i which is greater than C_i; since $CR > \mu$, there must be at least one job with $d_{im} > C_i$), and move all the jobs j,..., n forward in the time by

$\min\{gap, \min_{i=j,\ldots,u, E_i>0} E_i\}$. Update the earliness and tardiness of the moved jobs, set $k \leftarrow j$, $CR \leftarrow 0$ and go to step 4.

(d) If $CR > \mu$ and $u \neq n$, calculate $gap \leftarrow \min_{j \le i \le u, d_{im} > C_i} \{d_{im} - C_i\}$ (where d_{im} is the first due date of job i which is greater than C_i; since $CR > \mu$, there must be at least one job with $d_{im} > C_i$), and move all the jobs $j, \ldots, u$ forward in the time by $\min\{diff, gap, \min_{i=j,\ldots,u, E_i>0} E_i\}$. Update the earliness and tardiness of the moved jobs, set $k \leftarrow j$, $CR \leftarrow 0$ and go to step 3.

(e) If $CR \le \mu$ and $k = n$, stop.

In step 5(b), since the cost increases by moving jobs $j, \ldots, k$ forward, we stop considering those jobs for idle time insertion and continue starting with job $k+1$. Note that each time step 5(c) or 5(d) applies, it is not necessary that a job which was early becomes on-time, because we are taking *diff* and *gap* into consideration. At most we need $(n-1)*(m+1)$ iterations to make all jobs on time or tardy. In each iteration, there will be at most n jobs to consider, so the time complexity of the optimal timing algorithm is $O(n^2 m)$.

Theorem 3 *For a given sequence of jobs, algorithm* "IDLE TIME INSERT MUL DUE DATES" *inserts idle times optimally in between jobs when all costs are linear.*

Proof: Same as the proof of theorem 1. ❑

5.3 Modified GREEDY INSERT

Here we extend GREEDY INSERT for the problem at hand. Since in this problem each job can have multiple due dates and multiple tardiness penalties, we need to modify the way we sort the jobs. So, we tested a few sorting procedures for GREEDY INSERT. The following are some of the promising ones:

Sort 1

Sort the jobs in non-increasing order of β_{i1} / p_i.

Sort 2

Here we first find a range where a job is likely to complete and then take the average of all the tardiness penalties of that job corresponding to its due dates that fall in the range. Finally, we sort the jobs by non-increasing order of the ratio of that average penalty over the processing time. The details are as follows:

- Sort the jobs by non-decreasing r_i and make schedule to minimize $C_{\max}$. Assume that processing range is $[r_{\min}, C_{\max}]$.
- Assume each job j equally likely to finish in the interval $[r_i + p_i, C_{\max}]$.
- Compute "Expected tardiness penalty" of i :
  - Find all the due dates of i that are in the range of $[r_i + p_i, C_{\max}]$. We call them $d_{i1}, d_{i2}, \ldots, d_{ik}$ and the corresponding tardiness penalties $\beta_{i1}, \beta_{i2}, \ldots, \beta_{ik}$.
  - Expected tardiness penalty of $i = (\beta_{i1} + \beta_{i2} + \ldots + \beta_{ik}) / k$
- Sort the jobs by non-increasing order of $\dfrac{(\beta_{i1} + \beta_{i2} + \ldots + \beta_{ik}) / k}{p_i}$.

Sort 3

Sort the jobs in non-increasing order of $\beta_{i\max} / p_i$. where $\beta_{i\max}$ the max tardiness penalty of job i.

Sort 4

This sorting procedure is similar to sort 3 but here we scale each penalty by its corresponding range. Therefore we sort the jobs in non-increasing order of

$$\frac{\beta_{i0}(d_{i1} - (r_i + p_i)) + \beta_{i1}(d_{i2} - d_{i1}) + \ldots + \beta_{ik}(C_{\max} - d_{ik})}{C_{\max} - (r_i + p_i)}.$$

We start by sorting jobs using any one of the above sorting procedures and try to schedule each job at its first due date. If it does not overlap any previously scheduled jobs, we go on to the next one. Otherwise, we try placing that job at the beginning, then at the end, and then we try three ways of inserting it in between previously placed jobs close to its first due date, pushing other jobs earlier or later so that it fits. Finally we keep it in the position that results in the lowest total cost. One should keep in mind that no job can start before its release time. The details are as follows:

Algorithm GREEDY INSERT [GI]: Schedules n jobs each having possibly different release time and possibly different multiple due dates on a single machine in an attempt to minimize the total of flowtime, earliness, tardiness, and machine idle costs, which need not be linear.

1. Sort the jobs in non-increasing order by any one of the above mentioned procedures. Break ties by earliest due date first.
2. For $i = 1, 2, ..., n$ do:
   - If placing i so that it completes exactly at its first due-date does not interfere with any of the previously placed jobs, place it there.
   - Otherwise, select among the following five schedules the one with lowest total cost of all placed jobs:

   (a) Place i so that it comes first in the sequence of scheduled jobs and starts exactly at it's release time, with no inserted idle time between i and the next job. This may require pushing forward some previously scheduled jobs

   (b) Place i so that it comes last in the sequence of scheduled jobs, with no inserted idle time between i and the preceding job. Make sure that starting time of job i is at least greater than or equal to its release time.

   (c) If d_{ij} is in currently idle time, place i so that it completes at d_{ij} and push preceding jobs earlier so that they do not overlap and start before their release

times. Otherwise, find the closest idle time that occurs prior to d_{ij} and insert i there, pushing preceding jobs earlier if necessary. In both cases, if pushing preceding jobs earlier is not enough, succeeding jobs are pushed later.

(d) If $(d_{ij} - p_i)$ is in currently idle time, place i in that idle interval as late as possible if it fits and also starts at least by its own release time, or by pushing succeeding jobs later if it does not. Otherwise, find the closest idle time that occurs after $(d_{ij} - p_i)$ and insert i there, pushing succeeding jobs later if necessary.

(e) If d_{ij} is in currently idle time and fits in that idle interval, place it so that it starts at the start of the idle interval. If it does not fit, place it so that it completes at the end of the idle interval and push preceding jobs earlier to make it fit. Make sure that job i should not start before its own release time.

The five schedules tested in step 2 may not all be distinct ones. However, at least two options are tested: One placing the job early, and one placing it tardy. So although we are initially sorting the jobs according to their tardiness penalties only, the effect of the flowtime, earliness penalties and of the machine idle cost comes to play when choosing among the five schedules of step 2. GREEDY INSERT works in $O(n^2)$ and results in a schedule with idle times already inserted, through not necessarily in an optimal fashion.

5.4 Experimental Design

In order to test the performance of our heuristics and compare it to other procedures, we designed a 2^2 factorial experiment that takes T & R into account. We randomly generated the processing times of the jobs from a Discrete Uniform [1,30] distribution, the release time from a Discrete Uniform [0, $P/2$] distribution where $P = \sum_{i=1}^{n} p_i$, the first due date for all the jobs is generated as

$$d_{ij} = (r_i + U[P(1-T-R/2),\ P(1-T+R/2)])$$

and the next due date of the job is generated as

$$d_{ik} = d_{i(k-1)} + (U[P(1-T-R/2), P(1-T+R/2)]/n) \text{ for } k = 2,\ldots,m$$

where m is generated from a Discrete Uniform $[2, n]$ distribution. The first tardiness cost per unit time of the jobs is generated from Uniform $[0.5,5]P/n$ and the next tardiness cost per unit time is generated as $\beta_{i(k-1)} + (U[0.5,5]P/n)$. The earliness cost per unit time α_i form a Continuous Uniform [0.25, 0.75] β_i distribution, the work-in-process cost per unit time γ_i from a Continuous Uniform [0.1, 0.25]α_i distribution and the machine idle cost is generated from a Discrete Uniform [5, 25] distribution. We ran the experiment for instances with 10, 20, 30, 40 jobs per problem and tested different heuristic procedures against heuristics that have been introduced in the previous chapters for solving earliness, tardiness flowtime, and machine idle cost problems with different due dates with release times, and that can be easily modified to accommodate with multiple due dates. We generated a set of 800 problems for each instance. All the algorithms presented in this chapter were coded in PASCAL and run on a Pentium PC at 75 MHz.

5.5 Experimental Results

We use [GI1+] to designate the performance of GREEDY INSERT, which sorts jobs initially by sort 1 procedure, [GI2+] to designate the performance of GREEDY INSERT, which sorts jobs initially by sort 2 procedure, and [GI3+] to designate the performance of GREEDY INSERT, which sorts jobs initially by sort 3 procedure. We use [GI12+] to select the best solution obtained by [GI1+] and [GI2+]. Similarly [GI123+] to select the best solution obtained by [GI1+], [GI2+], and [GI3+]. The results are classified according to the value of tardiness and range factors. Some of the results obtained are presented in Table 5.1, Figures 5.2, 5.3 and remaining are tabulated in the Appendix-C

Some of the interesting results of our study are listed below:

- Performance of [GI12+], [GI123+] for loose due dates and small range is better than that of [API2], and very close to [API2] for wide range at loose and tight due dates.

- Performance of [GI12+] and [GI123+] is significantly better than [EDD], [SLK], and [E/S] for all due dates and range factors.

- Performance of GREEDY INSERT is greatly varied depending on initial sorting of jobs for all the cases and significantly noticeable for tight due dates at small range.

- CPU time required to solve the decomposed job shop cost problem is increased tremendously for [API2], [PI2] due to multiple due dates and multiple tardiness penalties compare to that of problems presented in the previous chapters.

- CPU time requirement for GREEDY INSERT is small and much less than that of [API2], [PI2].

- While Performance of GREEDY INSERT increases as problem size grows bigger for all cases except for tight due date and small range.

- Performance of [EDD], [SLK], and [E/S] is better for loose due dates than that of tight due dates.

- Performance of [PI2] is significantly better than all other heuristic presented here. But CPU time requirements is much higher than any other heuristic presented here.

- CPU time requirement for [EDD], [SLK], and [E/S] is smaller than GREEDY INSERT.

Table 5.1 Average performance of heuristics relative to API2:

No jobs	Due dates	Range	API2/EDD	API2/SLK	API2/ES	API2/GI 12+	API2/GI 23+	API2/PI2
10	Loose	Small	0.886	0.904	0.920	0.990	1.002	1.016
		Wide	0.924	0.935	0.954	0.986	0.993	1.007
	Tight	Small	0.796	0.748	0.827	0.948	0.979	1.045
		Wide	0.740	0.647	0.768	0.991	1.001	1.062
20	Loose	Small	0.894	0.903	0.912	0.995	0.998	1.023
		Wide	0.951	0.952	0.963	0.988	0.990	1.005
	Tight	Small	0.810	0.779	0.829	0.941	0.964	1.052
		Wide	0.741	0.683	0.780	0.977	0.992	1.082
30	Loose	Small	0.897	0.901	0.908	1.012	1.012	1.033
		Wide	0.956	0.959	0.964	0.995	0.997	1.009
	Tight	Small	0.834	0.823	0.850	0.920	0.946	1.046
		Wide	0.750	0.749	0.792	0.976	0.994	1.058
40	Loose	Small	0.898	0.902	0.907	1.016	1.016	-----
		Wide	0.958	0.960	0.965	0.996	0.997	-----
	Tight	Small	0.835	0.838	0.855	0.897	0.925	-----
		Wide	0.705	0.685	0.736	0.972	0.991919	-----

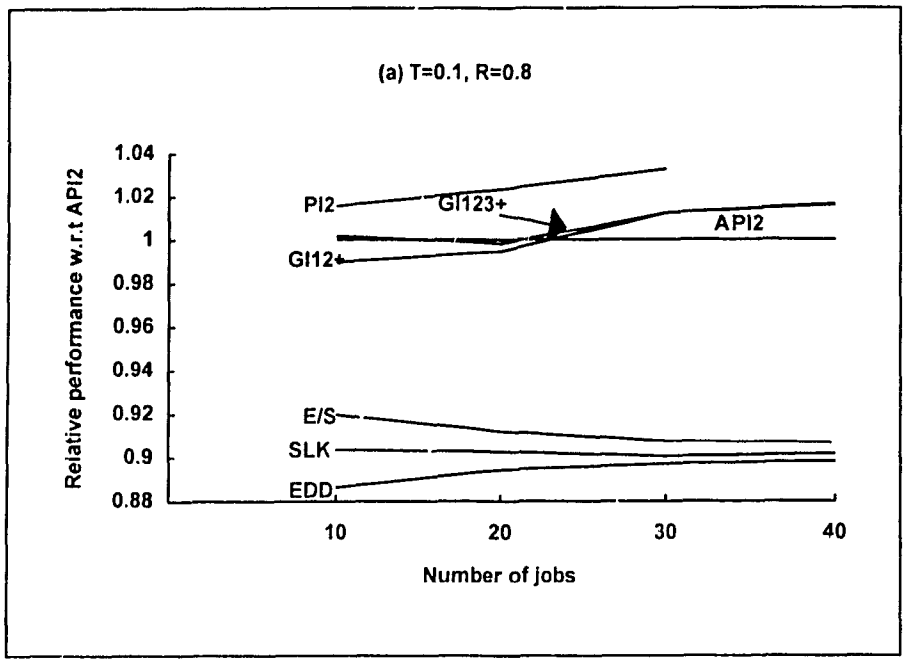


Figure 5.2(a) Loose due dates & Small range

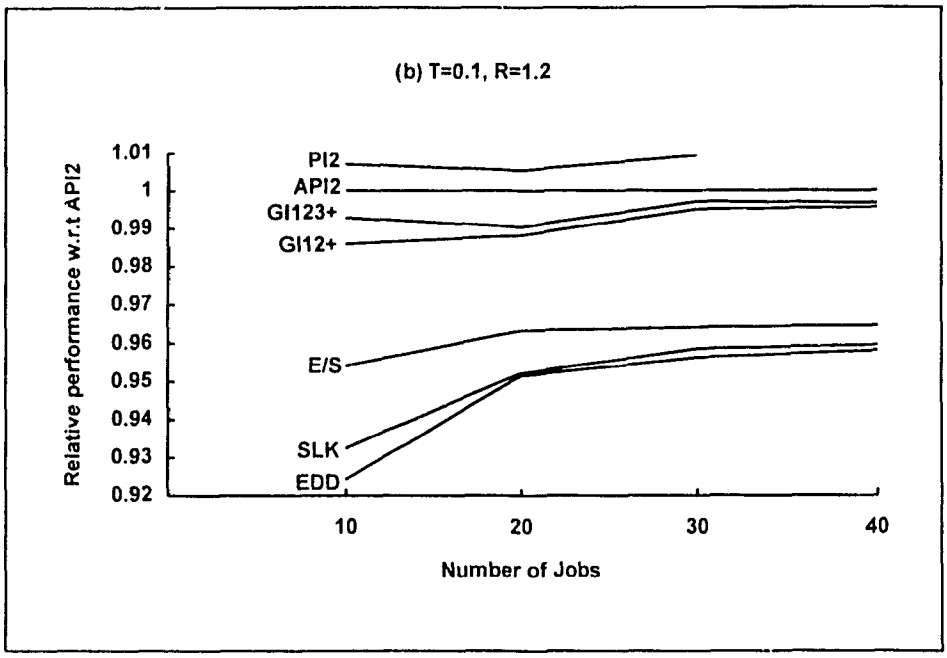


Figure 5.2(b) Loose due dates & wide range

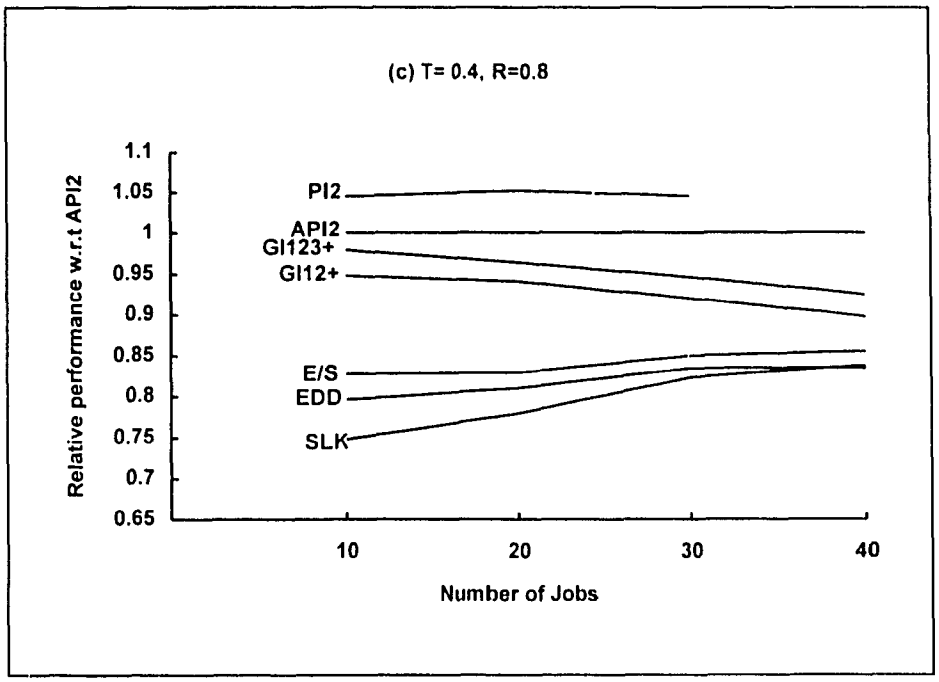


Figure 5.2(c) Tight due dates & Small range

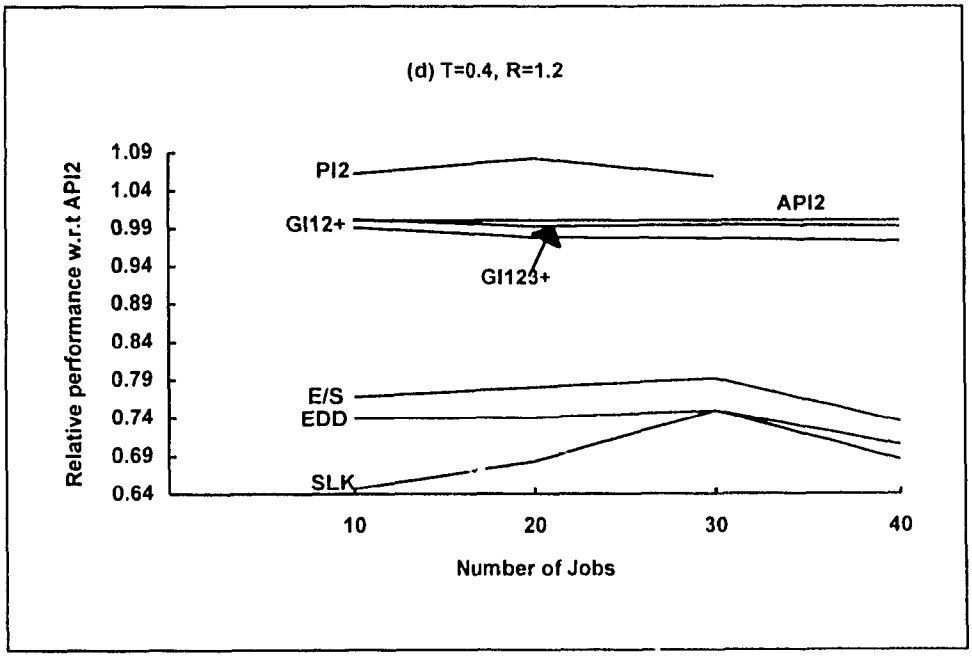


Figure 5.2(d) Tight due dates & Wide range

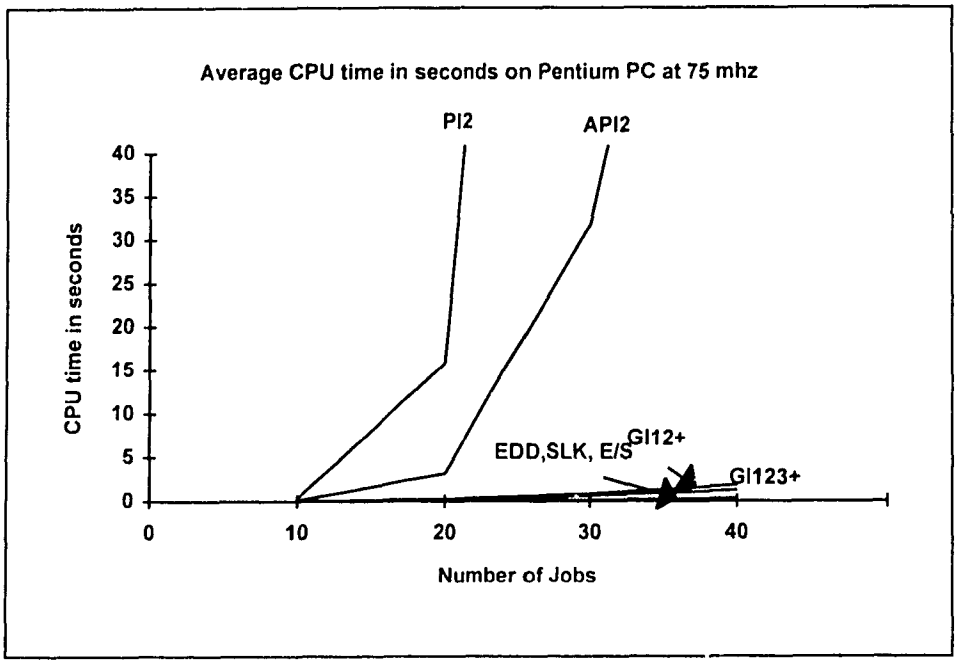


Figure 5.3 Average running time of algorithms

5.6 Conclusion

In this chapter, we designed & tested new heuristic procedures against other methods to solve the decomposed job shop cost problem to minimize the total of work-in-process, earliness, tardiness, and machine idle costs under different experimental conditions. The computational results showed that the new heuristics presented in this chapter will be an attractive alternative to solve big problem sizes efficiently with low computational cost. This kind of heuristic procedures are more important, since solving multiple single machine problems with reasonable accuracy with low computational cost solves complex job shop scheduling problems quickly with reasonable accuracy. The results obtained from new heuristic procedures are significantly varied depending upon the initial sorting procedure used. We recommend Sort 2 for initial sorting of jobs for [GI] because of its good performance. Here we did not use our version of tabu search because it is taking more time. So, for this problem we recommend usage of PI2 for small problem sizes and [GI12+] or [GI12+] for large problem sizes.

CHAPTER 6
Conclusion and Future Work

This thesis provided efficient heuristic procedures to solve a single machine problems to minimize work-in-process, earliness, tardiness, and machine idle costs where $\beta_i > \alpha_i > \gamma_i$. On the way to solve the decomposed job shop cost problem, this research also provided efficient heuristic procedures to minimize total cost on single machine scheduling problems with or without release times. The significance of machine idle cost was well demonstrated by taking that into the model to respond to the future demand in JIT manufacturing and also used to avoid slack schedules in solving job shop scheduling problems.

Generally it is not easy to obtain precise figures for flowtime, earliness, tardiness, and machine idle costs. Added to the fact that these problems are in a class of problems that are known to be computationally difficult which make heuristics such as |Gl| a logical choice. Whenever costs are not directly available, use relative importance to give weights.

Some of the important results are:

- Tardiness and Range factors have significant effect on the performance of heuristic procedures. Relative magnitude of work-in-process, earliness, tardiness, and machine idle costs have no significant effect on the relative performance of heuristic procedures.

- It is observed from the results that optimal insertion of idle times after each step of all search procedures will give significantly better results than optimal insertion of idle times at the end of the search.
- Tabu Search is recommended to solve small problem sizes (20 jobs). As problem size becomes larger, we highly recommend to use the GREEDY-INSERT heuristic to obtain a good solution with low computational cost. GREEDY INSERT is quick and performs very well on average, significantly better than the dispatching heuristics we tested. This makes [GI] an ideal heuristic procedure that provides perfect trade off between good solution and computational cost.
- To solve job shop scheduling problem, shifting bottleneck procedure proposed by Amiouny (1995) requires solving multiple single machine problems, therefore we recommend to use [GI] at early stages to get good solution in reasonable time, and use [API2] or Tabu search towards the end, if the computational power allows it for the problem.
- We recommend to choose heuristics based on the value of tardiness factor and due date range. For instance, it is highly advisable to choose [GI] to solve single machine problem with multiple due date for all cases except for the case with tight due dates and small range at large problem sizes.

Future research can focus on solving job shop problems to minimize total cost in JIT manufacturing with the help of heuristics which we developed in this thesis. We strongly believe that GREEDY INSERT is an efficient tool in solving complex job shop problems, which can be solved by solving multiple single machine job shop problems,

because of its low computational cost with reasonable accuracy. It will be interesting to use [GI] to solve job shop problems and try to compare it with other procedures. Since no other work done on the specific single machine problem with multiple due dates, others can use our results as benchmark.

Inspite of all the work done so far, the problem of minimizing total costs in job shop scheduling is likely to remain as a challenge to researchers for a long time to come.

References

1. Abdul-Razaq T. S. and C. N. Potts, "Dynamic programming state-space relaxation for single-machine scheduling", Journal of Operational Research Society, Vol. 19, No. 2, PP. 141-152, 1988.

2. Adams, J., Balas, E. and D. Zawack, "The shifting bottleneck procedure for job shop scheduling", Management Science, Vol. 34, No. 3, PP. 391-401, March 1988.

3. Adil, G. K., Rajamami, D. and D. Strong, " A mathematical model for cell formation considering investment and operational costs", European Journal of Operational Research, vol. 69, PP. 330-341, 1993.

4. Ahmadi, R. and U. Bagchi, "Single machine scheduling to minimize earliness subject to dead lines", Working paper 86/86-4-17, Department of management, University of Texas, Austin, 1986a.

5. Ahmadi, R. and U. Bagchi, "Just-In-Time scheduling in single machine systems", Working paper 85/86-4-21, Department of management, University of Texas, Austin, 1986b.

6. Amiouny, S.V., "A shifting bottleneck procedure for minimizing earliness/tardiness costs in job shop scheduling", Working Paper, Concordia University, Montreal, 1995.

7. Ashour, S., "Sequencing theory", New York: Springer-verlag, 1972.

8. Bagchi, U., "Scheduling to minimize Earliness and Tardiness penalties with a common due date", Working paper, Department of management, University of Texas, Austin, 1985.

9. Bagchi, U., Chang, Y. and R. Sullivan, "Minimizing absolute and squared deviation of completion times with different earliness and tardiness penalties and a common due date", Naval research logistics quarterly, Vol. 34, PP. 739-751, 1987.

10. Bagchi, U., Sullivan, R. and Y. Chang, "Minimizing mean absolute deviation of completion times about a common due date", Naval research logistics quarterly, Vol. 33, PP. 227-240, 1986

11. Bagchi, U., Sullivan, R. and Y. Chang, "Minimizing mean squared deviation of completion times about a common due date", Management Science, Vol. 33, PP. 894-906, 1987

12. Bahram, A., "Numerical methods for single machine scheduling with non-linear cost functions to minimize total cost", Journal of Operational Research Society, Vol. 44, No. 2, PP. 125-132, 1993.

13. Bahram, A. and D. Rosa, "A note on the V-shaped property in one - machine scheduling", Journal of the Operational Research Society, vol. 46, PP. 128-132, 1995

14. Baker, K. R., " Introduction to sequencing and scheduling", New York: John Wiely, 1974.

15. Baker, K. R. and G. D. Scudder, "Sequencing with earliness and tardiness penalties. A review", Operations Research, Vol. 38, No. 1, PP. 22-36, Jan-Feb. 1990.

16. Baker, K. R. and A. Chadowitz, "Algorithms for minimizing earliness and tardiness penalties with a common due date", Working paper No. 240, Amos Tuck School of Business Administration, Dartmouth College , Hanover, N.H, 1989.

17. Baker, K. R. and G. Scudder, "On the assignment of optimal due dates", Journal of Operational Research Society, Vol. 40, PP. 93-95, 1989.

18. Bector, C., Gupta, Y. and M. Gupta, "Determination of optimal common due date and optimal sequence in a single machine job shop", International Journal of Production research, Vol. 26, PP. 613-628, 1988

19. Bellman, Esogbue, A.O. and I. Nabeshima, "Mathematical aspects of scheduling & applications", International series in modern applied mathematics and computer science, Vol. 4, Pergamon Press1982.

20. Buxey, G., "Production scheduling: Practice and theory", European Journal of Operational Research, vol. 39, PP. 17-31, 1989.

21. Ceyda O. and C. Dincer, "Single machine Earliness-Tardiness scheduling problems using the equal -slack rule", Journal of Operational Research Society, Vol. 45, No. 5, PP. 589-594, 1994.

22. Chae, Y. L., "Genetic algorithms for single machine job scheduling with common due date and symmetric penalties", Journal of the Operations Research Society of Japan, Vol. 37, No. 2, PP. 83-95, June 1994.

23. Chae, Y. L. and J. Y. Choi, "A genetic algorithm for job sequencing problems with distinct due dates and general Early - tardy penalty weights", Computers and Operations research, Vol 22, No. 8, PP. 857-869, 1995.

24. Chae, Y. L. and S. J. Kim, "Parallel genetic algorithms for the earliness-tardiness job scheduling problem with general penalty weights", Computers ind. Engng, Vol. 28, No. 2, PP. 231-243, 1995.

25. Chand, S. and H. Schneeberger, "Single machine scheduling to minimize weighted earliness subject to tardy jobs", European Journal of Operational Research, vol. 34, PP. 221-230, 1988.

26. Chang, S. S. and U. G. Joo, "A single machine scheduling problem with earliness/tardiness and starting time penalties under a common due date", Computers and Operations research, Vol. 19, No. 8, PP. 757-766, 1992.

27. Cheng, T. C. E., "A duality approach to optimal due-date determination", Engineering Optimization, Vol. 9, PP. 127-130, 1985.

28. Cheng, T. C. E., "A note on the common due-date assignment problem", Journal of Operational Research Society, Vol. 37, No. 11, PP. 1089-1091, 1986.

29. Cheng, T. C. E., "An algorithm for the CON due date determination and sequencing problem", Comp. Opns. Res., Vol. 14, PP. 537-542, 1987.

30. Cheng, T. C. E., "Optimal common due date with limited completion time deviation", Comp. Opns. Res., Vol. 15, PP. 91-96, 1988a.

31. Cheng, T. C. E., "Optimal constant due-date assignment and sequencing", Int. J. Systems Sci., Vol. 1988, Vol. 19, No. 7, PP. 1351-1354, 1988b.

32. Cheng, T. C. E. " A note on a partial search algorithm for the single-machine optimal common due-date assignment and sequencing problem", Comp. Opns. Res., Vol. 17, No. 3, PP. 321-324, 1990.

33. Cheng, T. C. E., and M. Gupta, " Survey of scheduling research involving due date determination decisions", European Journal of Operational Research, vol. 38, PP. 156-166, 1989.

34. Cheng, T. C. E., and Z.-L. Chen, "Parallel-machine scheduling problems with earliness and tardiness penalties", Journal of Operational Research Society, Vol. 45, No. 6, PP. 685-695, 1994.

35. Chung-Lun, L., Cheng T. C. E. and Z.-L. Chen, "Single-machine scheduling to minimize the weighted number of early and tardy agreeable jobs", Comp. Opns. Res., Vol. 22, No. 2, PP. 205-219, 1995.

36. Chung-Yee, L., Danusaputro, S. L. and C.-S Lin, "Minimizing weighted number of tardy jobs and weighted earliness-tardiness penalties about a common due date", Comp. Opns. Res., Vol. 18, No. 4, PP. 379-389, 1991.

37. Dauzere-Peres, S. and J.-B. Lasserre, "A modified shifting bottleneck procedure for job shop scheduling", International Journal of Production Research, Vol. 31, No. 4, 923-932, 1993.

38. Davis, J. and J. Kanet, "Single machine scheduling with a non regular convex performance measure, Working paper, Department of Management, Clemenson University, Clemenson, S. C, 1988.

39. De, P., Ghosh, J. and C. Wells, "A note on the minimization of mean squared deviation of completion times about a common due date", Management Science, Vol. 35, PP. 1143-1147, 1989a.

40. De, P., Ghosh J. and C. Wells, "Scheduling about a common due date with earliness and tardiness penalties, working paper, University of Dayton, Dayton, Ohio, 1989b

41. Eilon, S. and I. Chowdhury, "Minimizing waiting time variance in the single machine problem ", Management Science, Vol. 23, PP. 567-575, 1977.

42. Emmons, H., "Scheduling a common due date on parallel common processors", Naval Research Logistics quarterly, Vol. 34, PP. 803-810, 1987.

43. French, S., "Sequencing and scheduling: An introduction to the mathematics of the job shop", New York: John Wiley and Sons, 1982.

44. Fry, T. D., G Keong Leong, "A bi-criterion approach to minimizing inventory costs on a single machine when early shipments are forbidden", Comp. Opns. Res., Vol. 14, PP. 363-368, 1987.

45. Fry, T. D., Darby-Dowman, K. and R. Armstrong, "Single machine scheduling to minimize mean absolute lateness", Working paper, College of Business Administration, University of South Carolina, Columbia, 1988.

46. Fry, T. D., Armstrong, R. D. and J. H. Blackstone, "Minimizing weighted absolute deviation in single machine scheduling", IIE Transactions, vol. 19, No. 4, PP. 445-450, 1987a.

47. Fry, T. D., Leong, G. K. and T. R. Rakes, " Single machine scheduling: A comparison of two solution procedures", OMEGA Int. J. of Mgmt Sci., vol. 15, No. 4, PP. 227-282, 1987b.

48. Garey, M. R., Tarjan, R. E. and G. T. Wilfong, "One-processor scheduling with symmetric earliness and tardiness penalties", Mathematics of Operations Research, Vol. 13, No. 2, PP. 330-348, May 1988.

49. Glover, F., "Tabu Search: A Tutorial", INTERFACES, Vol. 20, PP. 74-94, 4 July-August 1990.

50. Gupta, J. N. D., "Economic aspects of production scheduling systems", Journal of Operations Research Society of Japan, Vol. 13, No. 4, PP. 169-193, March 1971.

51. Gupta, S. and J. Kyparisis, "Single machine scheduling research", OMEGA, Vol. 15, PP. 207-227, 1987.

52. Gupta, S. and T. Sen, "Minimizing a quadratic function of job lateness on a single machine", Engn. Costs Prod. Econ., Vol. 7, PP. 181-194, 1983.

53. Hall, N., "Single and multi-processor models for minimizing completion time variance", Naval Research Logistics quarterly, Vol. 33, PP. 49-54, 1986.

54. Hall, N. G. and M. E. Posner, "Earliness-tardiness scheduling problems, I: Weighted Deviation of completion times about a common due date", Operations Research, Vol. 39, No. 5. PP. 847- , Sept.-Oct. 1991.

55. Hall, N. G., Kubiak, W. and S. P. Sethi, "Earliness-Tardiness scheduling problems, II: Deviation of completion times about a restrictive common due date", Operations Research , Vol. 39, No. 5, PP. 847- , Sept.-Oct. 1991.

56. Helmut G. K., "Scheduling with monotonous earliness and tardiness penalties", European Journal of Operational Research, Vol. 64, PP. 258-277, 1993.

57. Herrmann, J. W. and C. Lee, "On scheduling to minimize earliness-tardiness and batch delivery costs with a common due date", European Journal of Operational Research, Vol. 70, PP. 272-288, 1993.

58. Hoogeveen, J. A., Oosterhout H. and S. L. Van De Velde, "New lower and upper bounds for scheduling around a small common due date", Operations Research, Vol. 42, No. 1, Jan.-Feb. 1994.

59. Hoogeveen, J. A. and S. L. Van De Velde, "Scheduling around a small common due date", European Journal of Operational Research, Vol. 55, PP. 237- 242, 1991.

60. Kanet, J., "Minimizing the average deviation of job completion times about common due date", Naval Research Logistics Quarterly, Vol. 28, PP. 643-651, 1981a

61. Kanet, J., "Minimizing variation of flow time in single machine systems", Management Science, Vol. 27, PP. 1453-1459, 1981b.

62. Kim, Y. and C. A. Yano, "Minimizing mean tardiness and earliness in single-machine scheduling problems with unequal due dates", Naval Research Logistics Quarterly, Vol. 41, PP. 913-933, 1994.

63. Krieger, A. M. and M. Raghavachari, "V-shape property for optimal schedules with monotone penalty functions", Computers and Operations Research, Vol. 19, No. 6, PP. 533-534, 1992.

64. Lakshminarayan, S., Lakshmanan, R., Papineau, R. and R. Rochette, "Optimal single-machine scheduling with earliness and tardiness penalties", Operations Research, Vol 26, PP. 1079-1082, 1978.

65. Lawler, E. and J. Moore, "A functional equation and its applications to resource allocation and sequencing problems", Management Science, Vol. 16, pp. 77-84, 1969

66. Lawler, L. E., "A pseudopolynomial algorithm for sequencing jobs to minimize total tardiness, Annals of Discrete Mathematics, 1, 331-342, 1977.

67. Lenstra J. K., "Sequencing by enumerative methods", Mathematisch Centrum, Amsterdam, 1977.

68. Lenstra, J. K., Rinnooy Kan, A. H. G and P. Brucker, "Complexity of machine scheduling problems", Annals of Discrete Mathematics, 1, 331-342, 1977.

69. Maccarthy, B. L. and J. Liu, "Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling", International Journal of Production research, Vol. 31, No. 1, PP. 59-79, 1993.

70. Mittenthal, J. and M. Raghavachari, "Stochastic single machine scheduling with quadratic early-tardy penalties", Operations Research, Vol. 41, No. 4, July-Aug. 1993.

71. Morton, T. E. and D. W. Pentico, "Heuristic scheduling systems with applications to production systems and project management", New York: ETM Wiley series in engineering & technology management, 1993.

72. Ow, P. S. and T. E. Morton, "Filtered beam search in scheduling", International Journal of Production Research, Vol. 26, No. 1, PP. 35-62, 1988.

73. Ow, P. S. and T. E. Morton, "The single machine early / tardy problem", Management Science, Vol. 35, No. 2, Feb. 1989.

74. Panwalkar, S., Smith, M. and A. Seidmann, "Common due date assignment to minimize total penalty for the one machine scheduling problem", Operations Research, Vol. 30, PP. 391-399, 1982.

75. Parthasarati, D., "Common due date scheduling problem with separate earliness and tardiness penalties", Computers and Operations Research, Vol. 20, No. 2, PP. 179-184, 1993.

76. Prabuddha, D., Ghosh, J. B. and C. E. Wells, "Scheduling to minimize weighted earliness and tardiness about a common due-date", Computers and Operations Research, Vol. 18, No. 5, PP. 465-475, 1991.

77. Prabuddha, D., Ghosh, J. B. and Charles E. Wells, "On the general solution for a class of early/ tardy problems", Computers and Operations Research, Vol. 20, No. 2, PP. 141-149, 1993.

78. Quaddus, M., " A generalized model of optimal Due-date assignment by linear programming", Journal of Operational Research Society, Vol. 38, PP. 353-359, 1987a.

79. Quaddus, M., " On the duality approach to optimal due date determination and sequencing in a job shop", Engineering Optimization, Vol. 10, PP. 271-278, 1987b.

80. Ragavachari, M., "A V-shape property of optimal schedule of jobs about a common due date", European Journal of Operational Research, Vol. 23, PP. 401-402, 1986.

81. Ragavachari, M., "Scheduling problems with non-regular penalty functions - A Review", Opsearch, Vol. 25, PP. 141-164, 1988.

82. Raman, N. and F. B. Talbot, "The job shop tardiness problem: A decomposition approach", European Journal of Operational Research, Vol. 69, PP. 187-199, 1993

83. Rinnooy Kan, A. H. G., "Machine scheduling problems: Classification, Complexity and Computations", Martinus Nijhoff / The Hague, 1976.

84. Rinnooy Kan, A. H. G., Lageweg, B. J. and J. K. Lenstra, "Minimizing total costs in one-machine scheduling", Operations Research, Vol. 23, No. 5, PP. 908-927, Sept.-Oct. 1975.

85. Sen, T., and S. Gupta, " A state-of-the-art survey of static scheduling research involving due dates", OMEGA, Vol. 12, No. 1, PP. 62-76, 1984.

86. Sidney, J., "Optimal single-machine scheduling with earliness and tardiness penalties", Operations Research, Vol. 25, PP. 62-76, 1977.

87. Steve Davis, J. and J. J. Kanet, "Single-machine scheduling with early and tardy completion costs", Naval Research Logistics Quarterly, Vol. 40, PP. 85-101, 1993.

88. Sundararagavan, P. and M. Ahmed, "Minimizing the sum of absolute lateness in single-Machine and multi machine scheduling", Naval Research Logistics Quarterly, Vol. 31, PP. 325-333, 1984.

89. Suresh, C. and D. Chhajed, "A single machine model for determination of optimal due dates and sequence", Operations Research, Vol. 40, No. 3, PP. 596-602, 1992.

90. Surya, D. L. and S. Ramaswamy, "Earliness-tardiness scheduling problems with common delivery window", Operations Research Letters, Vol. 15, PP. 195-203, 1994.

91 Szware, W., "Minimizing absolute lateness in single machine scheduling with different due dates", working paper, University of Wisconsin, Milwaukee, 1988.

92. Szware, W., "Single machine scheduling to minimize absolute deviation of completion times from a common due date", Naval Research Logistics Quarterly, Vol. 36, PP. 663-673, 1989.

93. Szware, W., "Adjacent orderings in single-machine scheduling with earliness and tardiness penalties", Naval Research Logistics Quarterly, Vol. 40, PP. 229-243, 1993.

94. Vani, V. and M. Ragavachari, "Deterministic and random single machine scheduling with variance Minimization", Operations Research, Vol. 35, PP. 111-120, 1987.

95. Ventura, J. A. and M. X. Weng, "An improved dynamic programming algorithm for the single machine mean absolute deviation problem with a restrictive common due date", Operations Research Letters, Vol. 17, PP. 149-152, 1995.

96. Vonderembse, M. A. and G. P. White, " Operations Management Concepts, Methods, and Strategies", West Publishing Company, 1991.

97. Weng, M. X. and J. A. Ventura, "A note on single machine scheduling for minimizing total cost with identical, asymmetrical earliness and tardiness penalties", International Journal of Production Research, Vol. 32, No. 11, PP. 2725-2729, 1994.

98. Weng, M. X., and J. A. Ventura, "A quadratic integer programming method for minimizing the mean squared deviation of completion times", Operations Research letters, Vol. 15, PP. 205-211, 1994.

99. Wilhelm, S , "The job shop tardiness problem: A corrected model", European Journal of Operational Research, Vol. 79, PP. 549-550, 1994.

100. Yano C. and K. Yeong-Dae, "Algorithms for a class of single-machine weighted tardiness and earliness problems", European Journal of Operational Research, vol. 52, PP. 167-178, 1991.

101.Zheng, W. -X., Nagasawa, H., and Nishiyama, N, "Single-machine scheduling for minimizing total cost with identical, asymmetrical earliness and tardiness penalties", International Journal of Production Research, Vol. 31, PP. 1611-1620, 1993

APPENDIX-A

Experimental Results for the $n/1/FETM$ Problem

Table A.1 Average total cost is tabulated at T=0.1 & R=0.8 for the $n/1/FETM$ problem

Number of Jobs	EDD	SPT	SLK	E / S	GI	GI+	API1	API2	PI1	PI2	TABU25	TABU100
10	9245.476	17309.2	9081.60	8926.02	8269.71	8155.57	8256.56	8117.98	7944.30	7956.191	7894.283	7875.91
20	31624.49	67675.4	31284.4	30960.0	28160.1	27909.3	28364.7	27772.6	27086.1	27109.93	27038.15	26924.35
30	67225.84	148014	66915.2	66443.7	59845.1	59471.4	60668.2	59341.7	57704.14			
40	112109.2	242014	111933	111255	99005.6	98525.7	100853	98522.6	95424.28			
50	174130.3	------	173632	172995	152668	152042	157070	------	147545	------	------	------

Table A.2 Average total cost is tabulated at T=0.1 & R=1.2 for the $n/1/FETM$ problem

Number of Jobs	EDD	SPT	SLK	E / S	GI	GI+	API1	API2	PI1	PI2	TABU25	TABU100
10	7994.122	18767.8	7944.8	7815.89	7800.61	7643.74	7500.313	7457.782	7395.905	7399.067	7353.713	7347.716
20	26514.45	75098.5	26440.3	26179	25849.4	25615.2	25208.8	25036.5	23217.0	23194.64	23051.8	22927.88
30	55679.5	162393	55415.9	55157.2	54535.7	54171.1	53428.8	53124.9	52752.51			
40	91358.4	256234	91238.4	90794.7	89686.7	89253.3	88081.5	87571.4	86820.89			
50	140266.9	------	140122	139660	137964	137391	135695	------	133957	------	------	------

Table A.3 Average total cost is tabulated at T=0.4 & R=0.8 for the $n/1/FETM$ problem

Number of Jobs	EDD	SPT	SLK	E / S	GI	GI+	API1	API2	PI1	PI2	TABU25	TABU100
10	8457.0	16240.2	8914.8	8299.01	7273.37	7139.4	6883.66	6918.38	6603.47	6602.598	6538.571	6527.038
20	26049.5	62768.8	26849.6	25697.2	22185.0	21932.5	21464.8	21370.1	20369.1	20363.59	20257.02	20189.33
30	52594.2	136661	53260.5	52071.2	44752.2	44386.3	44512.3	43864.1	41725.44		------	------
40	86936.7	223480	87744.2	86338.2	73098.0	72677.5	73439.9	72141.8	86820.89		------	------
50	133587	------	134181	132785	111348	110804	113740	------	104697	------	------	------

Table A.4 Average total cost is tabulated at T=0.4 & R=1.2 for the n / 1 / *FETM* problem

Number of Jobs	EDD	SPT	SLK	E / S	GI	GI+	API1	API2	PI1	PI2	TABU25	TABU100
10	8146.128	18333.9	8502.1	8039.46	7343.65	7216.33	6775.37	6877.44	6652.18	6637.362	6599.486	7875.91
20	24088	71927.5	25101.5	23896.3	21561.0	21316.0	20364.4	20692.9	18589.2	18648.44	18475.78	26924.35
30	46850.8	154661	48314.4	46637.3	42653	42241.0	40432.8	41174.4	41725.44		------	------
40	73201.7	241930	74776.6	72909.7	67232.5	66722.9	64312	65262.2	68389.27		------	------
50	109826.1	------	111795	109583	101632	100948	97664.2	------	95974.8	------	------	------

Table A.5 CPU time on Pentium PC, 75 mhz for the n / 1 / *FETM* Problem:

Number of Jobs	EDD	SPT	SLK	E / S	GI	GI+	API1	API2	PI1	PI2	TABU25	TABU100
10	0	0	0	0.00125	0.011719	0.013281	0.005469	0.014063	0.026563	0.041406	0.658281	2.936
20	0	0	0.00213	0.00399	0.03203	0.03281	0.05016	0.16688	0.605	1.437813	10.9625	46
30	0.001563	0.001235	0.004688	0.005125	0.035938	0.048438	0.2575	1.1875	4.1875	------	------	------
40	0.004688	0.003125	0.00625	0.009136	0.067188	0.076563	0.73875	3.715313	10.2	------	------	------
50	0.007813	0.00780	0.00928	0.01744	0.14281	0.15141	1.41922	9.88797	21.8117	120.2319	------	------

APPENDIX-B

Experimental Results for the $n/1/r_i/FETM$ Problem

Table B.1 Average total cost is tabulated at T=0.1 & R 0.8 for the $n/1/r_i/FETM$ problem

Number of jobs	EDD	SLK	E / S	GI	GII	API2	PI2	TABU25
10	8968.7	8909.1	8728.9	8218.7	8132.3	8004.4	7939.1	7866.3
20	29152	29237	28734	26682	26484	26333	25874	25793
30	62007	61864	61401	56391	56031	56133	54905	54861
40	107689	107446	106841	96827	96212	96847	94258	------
50	162686	162281	161621	145025	144386	146063	141781	------
75	356056	355871	354874	317862	316815	322272	------	------

Table B.2 Average total cost is tabulated at T=0.1 & R =1.2 for the $n/1/r_i/FETM$ problem

Number of jobs	EDD	SLK	E / S	GI	GII	API2	PI2	TABU25
10	8210.1	8147.9	8022.1	7918.9	7775.6	7624.4	7573.4	7553.3
20	25831	25748	25517	25075	24852	24601	24437	24370
30	54643	54622	54247	53188	52816	52442	52134	52094
40	92560	92457	92056	90067	89562	88820	88226	------
50	138723	138626	138143	135246	134600	133921	133049	------
75	302307	302261	301614	296142	295185	294219	------	------

Table B.3 Results are shown below at T=0.4 & R = 0.8 for the $n/1/r_i/FETM$ problem

Number of jobs	EDD	SLK	E / S	GI	GI+	API2	PI2	TABU25
10	7379.8	7515.3	7031.8	6454.8	6338.1	6002.1	5882.5	5794
20	21899	22387	21553	19854	19596	18937	18478	18400
30	45837	45781	45246	40925	40453	39894	38703	38649
40	79484	79239	78617	69863	69056	68659	65996	------
50	120330	119979	119268	104494	103578	103557	99282	------
75	260594	260454	259448	226289	224987	227014	------	------

Table B.4 Results are shown below at T=0.4 & R = 1.2 for the $n/1/r_i/FETM$ problem

Number of jobs	EDD	SLK	E / S	GI	GI+	API2	PI2	TABU25
10	7860.7	8308	7610.2	6640.9	6581.4	6324.1	6119.3	6089
20	21443	22147	20908	18836	18614	18242	17728	17655
30	41124	41478	40584	38265	37858	37149	36555	36493
40	70629	71472	69689	63850	63208	61939	60837	------
50	101580	101834	100370	94282	93393	92497	91133	------
75	214915	216323	213867	203368	201953	200179	------	------

Table B.5 CPU Time in seconds on Pentium PC, 66 mhz for the n | 1 | r_i | *FETM* problem

Number of jobs	EDD	SLK	E / S	GI	GI+	API2	PI2	TABU25
10	0.00168	0.00169	0.00263	0.01013	0.01013	0.03633	0.14964	1.124413
20	0.00313	0.00306	0.00581	0.03081	0.03269	0.34828	2.33625	17.3225
30	0.00444	0.0045	0.00875	0.06325	0.06444	1.5927	20.595	84.8325
40	0.00631	0.007	0.0125	0.1133	0.11631	4.925	70.5	------
50	0.009	0.00788	0.01781	0.1869	0.19023	12	212.5	------
75	0.01756	0.01675	0.03363	0.45973	0.4756	70.9346	------	------

Table B.6 Fastness of heuristics with respect to API2 for the n / 1 / r_i / *FETM* problem

Number of jobs	EDD	SLK	E / S	GI	GI+	API2	PI2	TABU25
10	21.5259	21.5259	13.8381	3.58765	3.58765	1	0.24275	0.03231
20	111.448	113.722	59.9183	11.3030	10.6547	1	0.14907	0.02011
30	358.918	353.933	182.023	25.1810	24.7170	1	0.07733	0.01878
40	780.198	703.571	394	43.4687	42.3428	1	0.06986	------
50	1333.33	1523.81	673.684	64.2055	63.0832	1	0.05647	------
75	4038.86	4234.9	2109.58	154.298	149.148	1	------	------

APPENDIX-C

Experimental Results for the $n/1/r_i/d_{im}/FETM$ Problem

Table C.1 Average total cost is tabulated at T=0.1 & R=0.8 for the $n/1/r_i/d_{im}/FETM$ problem

Number of Jobs	EDD	SLK	E/S	API2	PI2	GI1	GI1+	GI2	GI2+	GI3+	GI12+	GI123+
10	8799.7	8631.68	8481.1	7799.57	7679.97	8003.99	7910.57	8066.55	7960.62	8241.66	7880.8	7786.5
20	31190.4	30907.3	30584.5	27894.4	27261.9	28479.1	28173.1	28853.7	28440.5	28896.5	28045.7	27947.1
30	64999.8	64751.1	64243.6	58305.5	56470.5	58153.5	57712.6	58855.7	58300.5	59347.4	57597.5	57596.5
40	108918	108508	107893	97838.2	------	97000.6	96360	98268.3	97533.3	99228.1	96295.4	96258.2

Table C.2 Average total cost is tabulated at T=0.1 & R=1.2 for the $n/1/r_i/d_{im}/FETM$ problem

Number of Jobs	EDD	SLK	E/S	API2	PI2	GI1	GI1+	GI2	GI2+	GI3+	GI12+	GI123+
10	7980.44	7908.89	7730.41	7374.86	7324.02	7659.87	7526.64	7650.72	7514.67	7599.71	7480.59	7428.44
20	27040.1	27019.8	26705.5	25725.3	25591.5	26770.3	26470.7	26674.3	26255.0	26359.1	26032.6	25975.3
30	55830	55696.5	55373.7	53385.5	52891.2	54244.8	53825.6	54460.7	53966.2	54084.9	53648.0	53537.6
40	92905.0	92761.5	92275.3	89009 5	------	90159.3	89520.5	90879.3	90176.3	90410.9	89401.2	89302.5

Table C.3 Average total cost is tabulated at T=0.4 & R=0.8 for the $n/1/r_i/d_{im}/FETM$ problem

Number of Jobs	EDD	SLK	E/S	API2	PI2	GI1	GI1+	GI2	GI2+	GI3+	GI12+	GI123+
10	7636.23	8126.13	7345.41	6077.28	5814.69	7554.62	7108.39	7197.74	6704.17	6721.3	6413.24	6208.69
20	25851.1	26861.5	25263.1	20934.7	19896.6	27276.3	26161.4	24758.2	23528.8	23535.0	22258.2	21724.6
30	50809.7	51511.9	49891.6	42393.0	40549.9	59795.7	56400.3	51162.2	49300.0	49822.4	46096.7	44828.1
40	85116.5	84785.7	83065.1	71033.5	------	102404	98510.8	89163.9	85840.2	84185.6	79158.3	76834.3

Table C.4 Average total cost is tabulated at T=0.4 & R=1.2 for the $n/1/r_i/d_{im}/FETM$ problem

Number of Jobs	EDD	SLK	E/S	API2	PI2	GI1	GI1+	GI2	GI2+	GI3+	GI12+	GI123+
10	9330.36	10664.0	8984.96	6899.85	6496.80	7927.09	7631.81	7327.82	7135.02	7603.23	6993.60	6890.74
20	28908.7	31341.9	27442.8	21406.5	19787.3	24524.6	24051.1	23610.8	22907.9	24784.4	21903.1	21576.6
30	53104.1	53116.4	50289.5	39807.5	37615.5	46329.3	45271.9	43720.7	43821.7	44163.8	40779.8	40042.9
40	96037.0	98766.2	92042.8	67695.4	------	80484.0	78446.6	75589.9	74267.7	80963.6	69627.3	68246.9

Table C.5 CPU Time is given below for the $n/1/r_i/d_{im}/FETM$ problem

Number of Jobs	EDD	SLK	E/S	API2	PI2	GI1	GI1+	GI2	GI2+	GI3+	GI12+	GI123+
10	0.00188	0.00169	0.0035	0.06185	0.29068	0.007	0.00818	0.00656	0.00781	0.00775	0.01494	0.0225
20	0.01506	0.01463	0.03156	3.115	15.7225	0.06144	0.07056	0.05775	0.06593	0.06775	0.15666	0.21333
30	0.05576	0.05571	0.1158	31.75	204.25	0.2248	0.2607	0.2233	0.26013	0.26135	0.57	0.80378
40	0.14946	0.11346	0.30086	109.5	------	0.53548	0.62833	0.515	0.6291	0.674	1.2025	1.805

Glossary

Some of the basic terms used in this thesis are introduced below:

Job: A job is a unit of a product that must be processed on certain machine. An alternative name is a task or an operation (S. Ashour 1972).

Machine: A machine is a single device capable of performing a certain process. An alternative name is a facility, processor, or work center (S. Ashour 1972).

Job Release Time: A job release time is the time at which a job is released to the shop after it has been engineered. It is equivalent to the earliest time that the processing of the job could start. An alternative name is job ready time, or job arrival time (S. Ashour 1972).

Job Start Time: A job start time is the time at which a job starts processing on the machine (S. Ashour 1972).

Job Completion Time: A job completion time is the time at which a particular job is completed. It is equivalent to the completion date of this particular job (S. Ashour 1972).

Machine Idle Time: A machine idle time is the length of time for which particular machine is not utilized before performing an operation or task (S. Ashour 1972).

Just-In-Time (JIT): Can be used as a basis for planning and scheduling, yet is more properly viewed as a strategy for designing manufacturing systems that are responsive to customer requirements. Apply JIT forces a reexamination of operating philosophy. The JIT philosophy focuses on reducing lead times, reducing set-up times and improving product quality to minimize raw material, work-in-process and finished goods inventory (Mark and Gregory 1991).