SINGLE MACHINE SCHEDULING PROBLEMS: EARLY-TARDY PENALTIES

A THESIS
SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL ENGINEERING
AND THE INSTITUTE OF ENGINEERING AND SCIENCE
OF BİLKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Ceyda Oğuz
March, 1993

SINGLE MACHINE SCHEDULING PROBLEMS: EARLY-TARDY PENALTIES

A THESIS

SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BİLKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Ceyda Oğuz

By

Ceyda Oğuz

March 1993

TS
157.5
.038
1993
c.1
B1866

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

[illegible]

Assoc. Prof. Dr. Cemal Dinçer (Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

[illegible]

Prof. Dr. Halim Doğrusöz

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

[illegible]

Assoc. Prof. Dr. Mustafa Akgül

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Ömer S Benli

Assoc. Prof. Dr. Ömer Benli

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

[illegible]

Assoc. Prof. Dr. Suna Kondakçı

Approved for the Institute of Engineering and Science:

M. Baray

Prof. Dr. Mehmet Baray,
Director of Institute of Engineering and Science

Abstract

SINGLE MACHINE SCHEDULING PROBLEMS:
EARLY-TARDY PENALTIES

Ceyda Oğuz
Ph. D. in Industrial Engineering
Supervisor: Assoc. Prof. Dr. Cemal Dinçer
March 1993

The primary concern of this dissertation is to analyze single machine total earliness and tardiness scheduling problems with different due dates and to develop both a dynamic programming formulation for its exact solution and heuristic algorithms for its approximate solution within acceptable limits. The analyses of previous works on the single machine earliness and tardiness scheduling problems reveal that the research mainly focused on a restricted problem type in which no idle time insertion is allowed in the schedule. This study deals with the general case where idle time insertion is allowed whenever necessary. Even though this problem is known to be $\mathcal{NP}$-hard in the ordinary sense, there is still a need to develop an optimizing algorithm through dynamic programming formulation. Development of such an algorithm is necessary for further identifying an approximation scheme for the problem which is an untouched issue in the earliness and tardiness scheduling theory. Furthermore, the developed dynamic programming formulation is extended to an incomplete dynamic programming which forms the core of one of the heuristic procedure proposed.

A second aspect of this study is to investigate two special structures for the different due dates, namely Equal-Slack and Total-Work-Content rules, and to discuss computational complexity of the problem with these special structures. Consequently, solution procedures which bear on the characteristics of the special due date structures are proposed. This research shows that the total earliness and tardiness scheduling problem with Equal-Slack rule is $\mathcal{NP}$-hard but can be solvable in polynomial time in certain cases. Moreover, a very efficient heuristic algorithm is proposed for the problem with the other due date structure and the results of this part leads to another heuristic algorithm for the general due date structure.

Finally, a lower bound procedure is presented which is motivated from the structure of the optimal solution of the problem. This lower bound is compared with another lower bound from the literature and it is shown that it performs well on randomly generated problems.

Keywords: Deterministic Single Machine Scheduling, Minimizing Total Earliness and Tardiness, Computational Complexity Theory, Dynamic Programming, Heuristic Algorithms, Lower Bounds.

Özet

TEK MAKİNA ÇİZELGELEME PROBLEMLERİ:
ERKEN-GEÇ PENALTILARI

Ceyda Oğuz
Endüstri Mühendisliği Doktora
Tez Yöneticisi: Doç. Dr. Cemal Dinçer
Mart 1993

Tek makinalı farklı teslim tarihli toplam erken-geç çizelgeleme problemlerini analiz etmek ve problemin hem kesin çözümü için bir dinamik programlama formülasyonu hem de yaklaşık çözümü için kabul edilebilir sınırlar içinde sezgisel yordamlar geliştirmek bu çalışmanın ana içeriğini oluşturmaktadır. Tek makinalı erken-geç çizelgeleme problemleri üzerine daha önce yapılmış çalışmaların incelenmesi, araştırmaların başlıca, çizelgede boş zaman ilave edilmesine izin verilmeyen, kısıtlı bir problem üzerine yoğunlaştığını göstermektedir. Bu çalışma, gerekli olduğu zamanlarda boş zaman ilavesine izin veren genel modelle ilgilenmektedir. Tek makinalı erken-geç çizelgeleme problemleri için, $\mathcal{NP}$-zor olmalarına rağmen, dinamik programlama formülasyonu yoluyla eniyi çözüm veren algoritmalara ihtiyaç vardır. Böyle bir algoritma, problem için bir yaklaşık yöntem tanımlayabilmek için gereklidir ki bu erken-geç çizelgeleme kuramında hemen hiç dokunulmamış bir alandır. Bundan başka, geliştirilen dinamik programlama formülasyonu, önerilen sezgisel yöntemlerden birinin özünü oluşturan kısıtlandırılmış durum uzaylı bir dinamik programlama şekline dönüştürülmüştür.

Bu çalışmanın ikinci bir safhası farklı teslim tarihleri için iki özel yapının, Eşit-Boşluk ve Toplam-İş-İçeriği kurallarının, incelenmesi ve bu özel yapılarla problemin hesap karmaşıklığının tartışılmasıdır. Bu problemler için, özel teslim tarih yapılarının özelliklerine dayanan çözüm yöntemleri önerilmiştir. Eşit-Boşluk kurallı toplam erken-geç çizelgeleme probleminin $\mathcal{NP}$-zor olduğu ispatlanırken, problemin belirli durumlarda polinom zamanda çözülebildiği de gösterilmiştir. Bundan başka, ikinci teslim tarihi yapısı ile problem için çok etkin bir sezgisel yordam önerilmiş ve bu problemden elde edilen sonuçlar, genel teslim tarihli problem için bir başka sezgisel yordamın geliştirilmesine öncülük etmiştir.

Son olarak, problemin eniyi çözümünün yapısından kaynaklanan bir alt sınır yöntemi sunulmuştur. Bu alt sınır, literatürden bir başka alt sınır ile kıyaslanmış ve geliştirilen bu alt sınırın rassal yaratılan problemler üzerinde iyi performans gösterdiği gözlemlenmiştir.

Anahtar Sözcükler: Deterministik Tek Makina Çizelgelemesi, Toplam Erken-Geç Enazlanması, Hesap Karmaşıklığı Teorisi, Kesin Çözüm Algoritmaları, Dinamik Programlama, Sezgisel Yordamlar, Alt Sınırlar.

Acknowledgement

I would like to express my appreciation to all those who have contributed directly or indirectly to this dissertation. I am grateful to Assoc. Prof. Cemal Dinçer for his invaluable supervision and encouragement throughout the development of this thesis. I am indebted to him for his interest and belief in my work. Above all, I gained the experience of conducting independent research and I thank him for his contribution to that.

I debt special thanks to Prof. Halim Doğrusöz, Assoc. Prof. Mustafa Akgül, Assoc. Prof. Ömer Benli, Assoc. Prof. Suna Kondakçı for their valuable remarks and discussions on the subject.

I would like to thank Prof. Thomas L. Morin for his contribution to the *incomplete dynamic programming* part of the thesis. His valuable discussions and comments were particularly helpful and led my research in new directions. I would like to express my thanks to referees, who have provided valuable comments on the papers to be published from this dissertation, for investing their time carefully reading the papers.

I owe substantial thanks to Dr. Cemal Akyel who acquainted me with the fascinating world of scheduling theory. I much appreciate the discussions with him at the various stages of this study. His morale support and encouragement throughout this study is greatly acknowledged.

Last but not the least, my sincere thanks are due to my family for their continuous morale support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Scheduling finds its application in a wide range of area whenever the problem of "the allocation of resources over time to perform a collection of tasks" (Baker, 1974) arises. For example, it is possible to make an assignment of classes to the classrooms in academic institutions. As a classic problem, we may encounter with outpatient visits to doctors in the hospitals. Another example can be given from the computer systems as the processing of the independent jobs on the processors. In all these examples, the first items (classes, outpatient visits, independent jobs) are the collection of tasks and the second items (classrooms, doctors, processors) are the resources and we can extend these examples to encompass a larger variety of activities of everyday life.

We can easily argue that the tremendous research since the early 1950's on the theory of scheduling is induced by this obvious practical importance and hence, it is not surprising that an impressive and enormous amount of literature has been evolving. But a careful investigation of the literature manifests that the main motivation and stimulation behind the existence of the scheduling theory as an important area in the operations research is its practical relevance to production planning and computer scheduling problems.

In this study we confine ourselves to scheduling problems which arise in the manufacturing systems so that each task, called *job*, requires at most one resource, called *machine*, at a time. The problem of concern is to schedule jobs on machines of limited capacity and availability which is called *machine scheduling* problem. The result of solving this problem is a *schedule* which specifies for each job when and by which machine it is to be processed. The aim is to find a schedule that optimizes some performance measures. The rich assortment of machine environments, job characteristics, and optimality criteria give rise to multitudinous machine scheduling problems. Over the spectrum of these scheduling problems, this thesis is concentrated on the area of *deterministic machine scheduling.*

Although, the scheduling models addressed by researchers have become more and more complex in order to better reflect the real situations through the years, they still have certain restrictive assumptions. The crucial assumption that is common in these models is related with the optimality criteria to measure the quality of the *feasible* schedules. The vast majority of the literature on machine scheduling pertain to regular performance measures which are non-decreasing in each of the job completion times. However, this type of performance measures may not interpret the practice and there are many important occasions when non-regular performance measures apply. For example, if the aim is the conformance to the due dates of the jobs, then the common practice is to penalize only the jobs which are finished after their due dates, called *tardy* jobs, ignoring the consequences of the jobs that complete before their due dates, called *early* jobs. In order to reach an acceptable optimality criterion, one has to measure the quality of a schedule on a criterion that incorporates the penalties arise from both early and tardy jobs which leads to a non-regular performance measure.

An important drawback of considering scheduling problems with non-regular performance measures lies in the difficulty of finding an optimal solution. The difficulty arises because in some cases the insertion of idle time between jobs will be beneficial which enlarges the set of feasible schedules. Regarding these two classes of the performance measures, this thesis is restricted to non-regular performance measures. Considering the hardness of these models, we further restrict ourselves to the single machine environments. Indeed, this last restriction is not so meaningless from both practical and theoretical point of view. First of all, because of its structural simplicity, it is easy to visualize the interactions in the model. It is also possible to explain the nature of the differences among different solutions and their relationships to different performance measures. Furthermore, the solutions of the single machine scheduling models can be used in more complex scheduling models by either being a basis for their solution procedures or supplying approximate but practical solutions. From a practical point of view, in addition to the existence of many shops which are actually a single machine, there are cases in which large and complex shops behave as if they are single machine environments. Examples of the latter can be seen in the chemical industries. Besides, there are many shops with more than one machine but either there is a single machine dominating all other machines in terms of job density or there exists a bottleneck machine, hence, viewing the shop as a single machine environment is an acceptable approximation.

More specifically, this thesis focuses on the deterministic single machine scheduling problems with the optimality criterion that aggregates the tardiness and earliness into a single objective function. In this introductory chapter, we present an overall view of single machine scheduling earliness and tardiness problem. We define the single machine earliness and tardiness problem formally in Section 1.1 and then elaborate on the reasons for involving both earliness and tardiness penalties in the objective function. In Section 1.2, we give the notation used throughout this thesis to represent scheduling problems which is based on the classification of machine scheduling problems introduced by Lawler*et al.* (1989). In Section 1.3, we point out how machine scheduling problems fit into the broader framework of combinatorial optimization and give an informal introduction to the theory of computational complexity. With the help of this theory, it is possible to classify problems as easy or probably hard to solve.

Introductions to these fields necessarily have to be selective and concise: only those concepts that are relevant for the subsequent chapters are discussed; others are merely touched upon. For more elaborate introductions to the respective areas, we refer to Conway *et al.* (1967), Baker (1974), Lawler *et al.* (1989) for machine scheduling and to Garey and Johnson (1979) for computational complexity.

1.1 Problem Definition

In the single machine total earliness and tardiness problem, we are given a set $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ of n independent jobs to be processed on a single machine that can handle at most one job at a time without preemption. In an instance of this problem, a processing time $p_j \in \mathcal{Z}^+$, a ready time $r_j \in \mathcal{Z}^+$ on which job J_j becomes available for processing, a due date $d_j \in \mathcal{Z}^+$ by which job J_j should ideally be completed, and weights $w_j \in \mathcal{Z}^+$ and $v_j \in \mathcal{Z}^+$ indicating the relative importance of job J_j as being early and tardy, respectively, can be specified for each job J_j. We can also determine for every job J_j a target starting time $a_j = d_j - p_j$ by which job J_j should ideally be started. Given a processing order on the single machine, the starting time S_j, the completion time $C_j = S_j + p_j$, the tardiness $T_j = \max\{0, C_j - d_j\}$, the earliness $E_j = \max\{0, d_j - C_j\}$ and the lateness $L_j = C_j - d_j$ for each job J_j can be computed such that the capacity and availability constraints of the machine are not violated. In an instance of this problem, we will use $\mathcal{S}$ and $\mathcal{C}$ for the set of starting times and the set of completion times, respectively. The penalties will be given in the set $\mathcal{P}$ where, in this representation, the negative and the positive values denote the earliness and the tardiness of a job J_j, respectively. The quality of a schedule is measured in terms of the optimality criterion which is the scheduling cost incurred as a function of the earliness and the tardiness of each job J_j, stated as $g_j(E, T)$. The optimality criteria covered in this study involve the minimization of

$$\max\{\max_{1\leq j\leq n} E_j, \max_{1\leq j\leq n} T_j\}$$

or of

$$\sum g_j \in \{\sum(E_j + T_j), \sum(w_jE_j + v_jT_j)\}$$

where $\sum g_j = \sum_{j=1}^{n} g_j(E/T)$ with $g_j(E/T) = (E_j + T_j)$, and $(w_jE_j + v_jT_j)$, respectively.

In the analysis of the given scheduling problem, we will use the following additional notation:

- π and σ denote sequences of jobs. Furthermore, $\mathcal{J}_\pi$ and $\mathcal{J}_\sigma$ denote the set of jobs that form the sequences π and σ, respectively.
- $\pi(i)$ and $\sigma(i)$ denote the i-th job in the sequence. Hence, i is the position index and $i = 1, 2, \ldots, n$.
- $z(\sigma)$ denotes the value of the optimality criterion for the schedule of σ and $z(\sigma) = \sum_{J_j \in \mathcal{J}_\sigma}(E_j + T_j)$, unless otherwise stated.
- $\mathcal{E}$ ($\mathcal{E}_S$) represents the set of jobs that complete before the due date (target starting time), where $|\mathcal{E}|$ ($|\mathcal{E}_S|$) denotes the cardinality of $\mathcal{E}$ ($\mathcal{E}_S$).
- $\mathcal{E}'$ ($\mathcal{E}'_S$) represents the set of jobs that complete exactly on or before the due date (target starting time), where $|\mathcal{E}'|$ ($|\mathcal{E}'_S|$) denotes the cardinality of $\mathcal{E}'$ ($\mathcal{E}'_S$).
- $\mathcal{T}$ ($\mathcal{T}_S$) represents the set of jobs that start exactly on or after the due date (target starting time), where $|\mathcal{T}|$ ($|\mathcal{T}_S|$) denotes the cardinality of $\mathcal{T}$ ($\mathcal{T}_S$).
- $\mathcal{T}'$ ($\mathcal{T}'_S$) represents the set of jobs that complete after the due date (target starting time), where $|\mathcal{T}'|$ ($|\mathcal{T}'_S|$) denotes the cardinality of $\mathcal{T}'$ ($\mathcal{T}'_S$).

We illustrate these notions by two 5-job examples for different due date structures. In the first example, all jobs have the same due date which is called as a common due date, d. In the second example, all jobs have a different due date but all have a common target starting time, q. The data are given in Figure 1.1. An a)

j	1	2	3	4	5
p_j	3	4	5	6	7
d_j	17	17	17	17	17

b)

j	1	2	3	4	5
p_j	3	4	5	6	7
d_j	18	19	20	21	22

Figure 1.1: Data of each example

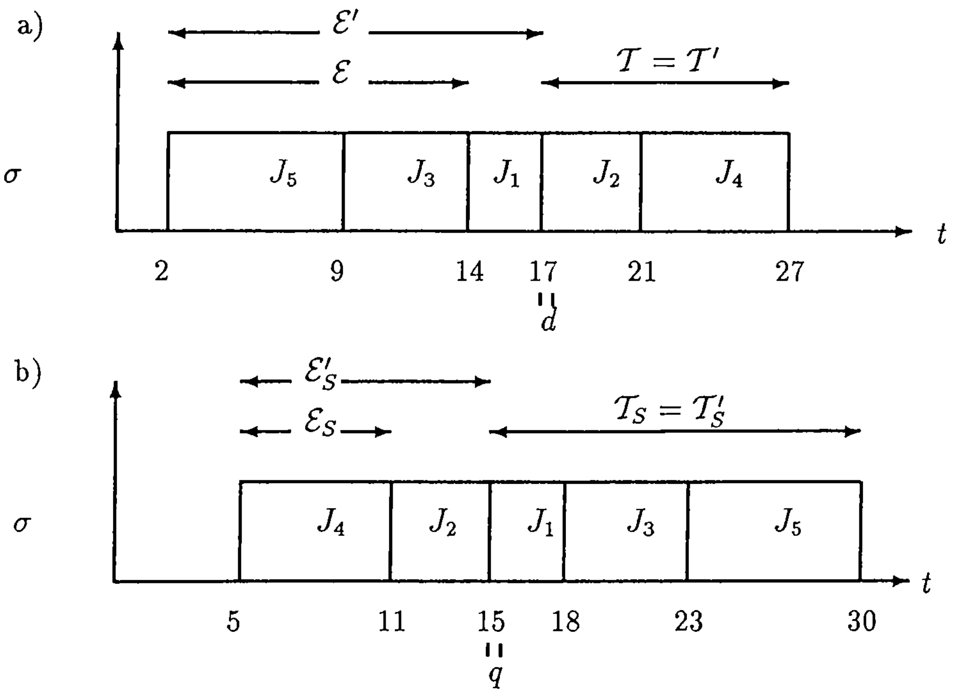


Figure 1.2: Schedule of σ for the example

arbitrary schedule of σ is represented in the *Gantt Chart* in Figure 1.2 for each example.

Scheduling problems consisting earliness and tardiness penalties in their objective function may have many applications in industry where both early and late completion of jobs from their due dates are costly and hence undesirable. For example, both earliness and tardiness penalties are in the nature of production environments such as having perishable products or applying Just-In-Time concept, since the deliveries should be coordinated with the manufacturing process steps due to the less adjustable delivery times than process steps.

The inclusion of an earliness cost in the objective function may represent the cost of completing a project early in PERT-CPM analyses, as suggested by Sidney (1977). Apart from these, we can give the example of scheduling a sequence of experiments that depend on predetermined external events such as the position of the sun as a natural application of earliness and tardiness penalties.

If we consider the completion of a job after its due date, it is common to incur costs due to the loss of the order and loss of customer goodwill. The cost of tardiness also includes customer dissatisfaction, contract penalties and potential loss of reputation. On the other hand, completion before the due date may lead to higher inventory costs, increases the danger of over-stocking in the event of order cancellation, and if goods are perishable, causes potential loss of usable production due to their deterioration. If it is assumed that once a job completes its processing, it is free to leave the system, then earliness will not be a problem. But in many cases, customers do not want to receive orders early since they will hold unnecessary inventory. This can cause the cash commitment to resources in a time frame earlier than needed.

Indeed, earliness costs have a different nature from tardiness costs because they are of an indirect nature. Early jobs tie up capital, take up scarce floor space, and generally indicate that resource allocation and utilization may have been less than optimal. In essence, the cost of earliness is a cost of inefficiency and represents an unproductive investment which implicitly incurs an opportunity cost. Since minimizing the earliness and tardiness penalties has important practical applications, the research on scheduling problems with earliness and tardiness penalties are growing rapidly.

1.2 Classification of Machine Scheduling Problems

Since there exists multifarious machine scheduling problems, we need a classification scheme to make them rapidly accessible and easy to refer to. We follow the notation and terminology of the classification scheme for deterministic machine scheduling problems as suggested by Lawler *et al.* (1989). In this notation each scheduling problem is represented by means of three parameters α, β, γ, where:

- α identifies the machine environment, such as single machine (1). We do not deliberate on the multi-machine environments, for which α has many different expressions, since our research deals with single machine environment.
- $\beta \subset \{\beta_1, \ldots, \beta_4\}$ identifies the job characteristics, which are defined as follows:
  - $\beta_1 \in \{pmtn, \emptyset\}$.
    $\beta_1 = pmtn$: *Preemption* is allowed: the processing of any operation may be interrupted and resumed at a later time.
    $\beta_1 = \emptyset$: No preemption is allowed.
  - $\beta_2 \in \{r_j, \emptyset\}$.
    $\beta_2 = r_j$: *Release dates* that may differ per job J_j are specified.
    $\beta_2 = \emptyset$: All $r_j = 0$.
  - $\beta_3 \in \{p_j = 1, \emptyset\}$.
    $\beta_3 = p_j = 1$: Each job J_j has a *unit processing requirement.*
    $\beta_3 = \emptyset$: All p_j are arbitrary nonnegative integers.
  - $\beta_4 \in \{d, d_j, \emptyset\}$.
    $\beta_4 = d$: A *common due date* is specified for each job J_j.
    $\beta_4 = d_j$: *Distinct due dates* are specified arbitrarily for each job J_j.

$\beta_4 = \emptyset$: *Due dates* are variables, the values of which have to be determined.

- γ identifies the optimality criterion of the scheduling problem, such as the total unweighted earliness and tardiness penalties ($\sum E_j + T_j$).

A three-field-notation scheme $\alpha|\beta|\gamma$ will be used to describe a machine scheduling problem. Hence, the problem of determining an optimal single machine schedule with minimum total earliness and tardiness penalties is denoted by $1|d_j|\sum g_j(E/T)$, following the above terminology.

1.3 Combinatorial Optimization

Machine scheduling problems belong to the area of *combinatorial optimization*. Combinatorial optimization involves problems in which we have to choose the best from a finite number of relevant (*feasible*) solutions over a combinatorial (*discrete*) set. For the total earliness and tardiness scheduling problems that are of concern, for instance, we can restrict ourselves to the $n!$ permutations of the n jobs. Indeed, the solution space is larger than this, since for each permutation (or sequence) there is more than one feasible schedule due to the inserted idle times before or between the execution of jobs. But the optimal schedule can be obtained easily once a sequence is specified (see Section 2.6.4).

The finiteness of the solution set by implying the effectiveness of the brute-force approach of *explicit enumeration* may be misleading. Since the optimal solution can be obtained by a straightforward method that generates all feasible solutions and select the best one with respect to its objective function. Unfortunately, however, the efficiency of this type of enumeration methods is far from being satisfactory for solving large scale problems of practical importance since the required effort to examine all schedules grows exponentially with the number of jobs.

We have therefore good reasons to search for faster algorithms. At this point a fundamental question is whether a problem is 'well solved' ('easy') or 'hard'. The distinction between easy and hard problems apparently involves the effort required to solve them to optimality. Since the effort grows with the size of the problem instance, it makes sense to express the effort as some function of this size. The *size of an instance* is defined as the number of symbols required to represent an instance and this depends on *encoding* that is the representation system we employ. Integers may be represented by an arithmetic system to some fixed base $B \geq 2$, in which case $\lceil \log_B n \rceil$ symbols are required to represent an integer n. If $B = 2$, then we have a *binary* encoding. Another system is a *unary* encoding. Under a unary encoding, integers are represented by a series of 1's, the length of which is equal to the value of the integer: to represent an integer n, we need n symbols.

The *time complexity* of an algorithm for a given problem is measured by an upper bound on the number of computation steps that the algorithm performs on any valid input, expressed as a function of the size of the input. If the size of the instance is measured by n, then the running time of an algorithm is expressed as $\mathcal{O}\ (f(n))$ if there are constants c and n_0 such that the number of steps for any problem instance with $n \geq n_0$ is bounded from above by $c f(n)$, for some function f.

An optimization problem is said to be easy if there exists an algorithm that solves the problem in time *polynomially* bounded in the input size. On the other hand, if any algorithm for the problem requires a complexity not bounded above by a polynomial in n, it is considered to be hard.

1.3.1 Computational Complexity

In discussing the complexity of a problem, it is sometimes more convenient to use the decision problem rather than the optimization problem. The decision variant of a scheduling problem is defined as the following question: given an instance of the problem and a threshold value y, does there exist a schedule with value no more than y? All easy decision problems constitute the class $\mathcal{P}$. This class is a subset of the class $\mathcal{NP}$, which, in the present context, contains all decision problems for which it is possible to check in polynomial time if the answer is 'yes' for a given schedule. A decision problem is said to be $\mathcal{NP}$-complete if it belongs to $\mathcal{NP}$ and if every problem in $\mathcal{NP}$ is *polynomially reducible* to it. A problem Π is said to be polynomially reducible to a problem Π' if and only if an arbitrary instance of Π can be solved by solving a corresponding instance of Π' that is constructed in time polynomially bounded in the size of Π. The optimization variant of an $\mathcal{NP}$-complete problem is called $\mathcal{NP}$-hard; these problems are at least as hard as all problems in $\mathcal{NP}$.

Coming to the different encoding schemes, the distinction between binary and unary encoding of the input is relevant for those problems that are $\mathcal{NP}$-complete under a binary encoding, but solvable in polynomial time under a unary encoding. An algorithm which is polynomial under a unary encoding, but not polynomial under a binary encoding, is called a *pseudo-polynomial time* algorithm. Problems that are $\mathcal{NP}$-complete under both encodings are called *strongly* $\mathcal{NP}$-complete. Problems are said to be *ordinarily* $\mathcal{NP}$-complete if they are $\mathcal{NP}$-complete under a binary encoding. For details concerning computational complexity, refer to Garey and Johnson (1979).

1.3.2 Optimization

Although optimization algorithms for hard combinatorial optimization algorithms are unavoidably enumerative in nature, the aim is still to develop algorithms that perform satisfactorily well on the average for instances of reasonable size. *Dynamic programming* and *branch-and-bound* are two major enumerative methods to solve hard combinatorial optimization problems.

Both dynamic programming and branch-and-bound aim at *implicit enumeration* of the solution space. For the application of dynamic programming, we need to identify some underlying principle of optimality. Application of the optimality principle may require both time and space that is not bounded by a polynomial in the length of the input. Nonetheless, dynamic programming based pseudo-polynomial algorithms may be very efficient.

Branch-and-bound algorithm mainly consists of a branching structure of the problem which generates feasible set of solutions for smaller subsets of the original set, and lower and upper bounding functions as well as a dominance relation that together constitute a bounding operation which is used to discard some subsets from further consideration. The performance of a branch-and bound algorithm is affected firstly by the strength of the lower bounds which restrict the growth of the feasible sets. Furthermore, the quality of the upper bounds, the dominance relations, and also the branching structure directly affect the efficiency of a branch-and-bound algorithm.

1.3.3 Approximation

It is obvious that an optimization algorithm for a hard combinatorial optimization problem will take exponential amount of time in the worst case. This leads to deal with a good *approximate* solution which can be obtained in a reasonable time and with less effort instead of finding an optimal solution. For this case, the main question is the trade-off between the quality of the approximate solution and the time spent to find it.

There are a number of popular techniques for designing approximation algorithms for the machine scheduling problems. For example applying *dispatching rules* which make use of priority functions that associate an urgency measure for each job is one of the simplest and widely used technique. The second class contains the approximation algorithms based upon the dynamic programming and rounding which is known as *incomplete dynamic programming*. The main idea is to consider a specific part of the state space instead of the entire state space. The last class of techniques that we point out contains the *local-search algorithms*.

These type of algorithms first generate an initial schedule and then adjust it somewhat in order to improve the objective function value. One of the most widely used procedure is to define for a given schedule of σ a *neighborhood* N_σ as the set of schedules that can be obtained from σ by carrying out a prespecified type of changing operations, such as *adjacent pairwise interchange.*

Globally, we can say that local-search algorithms are easy to develop and implement, and are known to produce excellent results.

1.4 Outline of this Thesis

The main aim of this thesis is to analyze the single machine total unweighted earliness and tardiness scheduling problems. Although there exists considerable research for the case where all the due dates are same, the literature is limited on the problems where all the jobs have distinct due dates. There is a lack of both optimizing and heuristic algorithms for total earliness and tardiness scheduling problems with distinct due dates. The purpose of this study is to develop computationally efficient optimizing algorithms and heuristic algorithms that solve the addressed problem effectively. Another purpose of this research is to investigate some special structures for the due dates and then either to show that the problem is $\mathcal{NP}$-hard or to provide polynomial algorithms for the problem.

This thesis is organized as follows. In Chapter 2, the characteristics of the total earliness and tardiness problems are analyzed and previous work on these problems are reviewed concentrating on the problem $1|d_j|\sum(E_j + T_j)$. The discussion on the approaches of the problem $1|d_j|\sum(E_j+T_j)$ concludes that there is a lack of both an exact algorithm and a heuristic procedure which solve this problem efficiently and effectively. One of the main chapters, Chapter 3, presents a detailed development of a dynamic programming formulation for the problem $1|d_j|\sum(E_j + T_j)$, including the extension of this formulation as an incomplete dynamic programming. This approach forms also the basis for the heuristic procedure developed and this heuristic procedure is given in Chapter 3, together with the computational results. Chapter 4 analyzes two special structures for the distinct due dates of the problem $1|d_j|\sum(E_j + T_j)$. In this chapter, it is shown that there exists two cases for the first structure. Chapter 4 first presents the equivalence of one of the cases to a polynomially solvable problem and then proves that the other one is $\mathcal{NP}$-hard. Chapter 4 also investigates the effect of a second special structure for the due dates on the problem $1|d_j|\sum(E_j + T_j)$. After providing some properties regarding to this special structure, Chapter 4 concludes with an efficient heuristic algorithm. The first part of Chapter 5 extends the results obtained in Section 4.2 and develops an alternative heuristic algorithm for problem $1|d_j|\sum(E_j + T_j)$. In the second part of Chapter 5, a lower bound procedure for problem $1|d_j|\sum(E_j + T_j)$ is presented and its effectiveness is tested on randomly generated problems. In Chapter 6, the significance and the importance of the results of this study and possible directions for future research are discussed.

Chapter 2

Review of Single Machine Earliness and Tardiness Problems

The subject of this thesis is to analyze the single machine total unweighted earliness and tardiness problems with distinct due dates. In this chapter, some of the early work on single machine total earliness and tardiness problems that is relevant to this research are briefly reviewed and analyzed. In the last section, Section 2.6.4, the main problem that we deal with is reviewed in detail which shows the deficiencies of the approaches on this problem.

In the following section, the characteristics and the basic definitions in the single machine earliness and tardiness problems are presented. Then, in Section 2.2 a classification of the earliness and tardiness problems is given. After giving the mathematical formulation of the problem in its most general form in Section 2.3, two special cases are stated in Section 2.4. Then, the problems that are categorized according to the classification given in Section 2.2 are reviewed considering their complexity results in Section 2.5 and Section 2.6.

2.1 Problem Characteristics

The objective function of the scheduling problems can be analyzed in two different classes. The first class consists of the one dimensional performance measures which are called *regular performance measures*. The second class is called as the *non-regular performance measures*. A schedule for the sequence σ is defined as a vector of job completion times $[C_1, C_2, \ldots, C_n]$. Then the performance measure can be denoted as:

$$z(\sigma) = z(C_1, C_2, \ldots, C_n)$$

Definition 2.1 [Baker, 1974] *A performance measure $z(\sigma)$ is regular if*

(a) the scheduling objective is to minimize $z(\sigma)$, and

(b) $z(\sigma)$ is non-decreasing in each of the completion times, C_j, in the schedule.

This definition is important because it is usually desirable to restrict attention to a limited set of schedules called a *dominant set*. For example, in the single machine scheduling problem, the set of permutation schedules is a dominant set for any regular performance measures. That is, preemption and inserted idle time will never lead to a better schedule than the best permutation schedule for any regular performance measure.

Most popular performance measures, such as mean flowtime, mean lateness, mean tardiness, and number of tardy jobs, are all regular performance measures. There are, however, many applications in which non-regular performance measures are more appropriate.

Definition 2.2 *A performance measure $z(\sigma)$ is non-regular if the property (b) does not hold in Definition 2.1.*

For example, the mean tardiness criterion has been a standard way of measuring conformance to due dates, although it ignores consequences of jobs completing early. Hence, if it is desirable to meet all due dates exactly, it is necessary to consider non-regular performance measures. In spite of the importance of non-regular performance measures, very little analytical work has been done in this area probably due to the difficulty of solving these type of problems. The difficulty arises because in some cases the insertion of idle time between jobs will be beneficial which may prevent the set of permutation or *non-delay* schedules to be a dominant set. In this case, *delay schedules* should be considered which increase the solution space due to the inserted idle time in the optimal solution.

Definition 2.3 [Fry, Armstrong and Blackstone, 1987] *A delay schedule is any schedule where the machine is intentionally held idle when it could begin processing a job.*

Theoretically, there are infinite number of delay schedules because arbitrary amounts of idle time can be inserted. However, this is not useful for a given sequence and a non-regular performance measure and we should restrict the feasible solution set to a finite number of schedules.

Definition 2.4 *A local shift is changing the completion time of a job in a feasible schedule while keeping feasibility without changing the sequence.*

Definition 2.5 [Davis and Kanet, 1992] *A semi-active schedule for a non-regular performance measure is one in which no local shift of a job will provide a schedule with an improved value of objective function.*

It is obvious from the above definition that semi-active schedules dominate the set of all schedules. Since it is possible to insert some idle time between the jobs in a schedule, we can talk about a group of jobs such that there exists some idle time before the first job and after the last job of the group, but not between any two of the jobs in this group. We can state this concept formally as follows:

Definition 2.6 *The blocks of the schedule are the maximal sets of jobs $J_{j_0}, J_{j_0+1}, \ldots, J_{j_1}$ that are scheduled without any idle time between any two of these jobs, i.e., $S_j + p_j = S_{j+1}$ $\forall$ $j_0 \leq j < j_1$, $S_{j_0-1} + p_{j_0-1} < S_{j_0}$ (or $j_0 = 1$), and $S_{j_1} + p_{j_1} < S_{j_1+1}$ (or $j_1 = n$, the last position to be considered).*

2.2 Classification of Earliness and Tardiness Problems

Diversity in the total earliness and tardiness literature stem from the generality of assumptions made about the due dates and the optimality criteria. The single machine total earliness and tardiness problems can be classified according to

A. the condition on the due date:

1. problems having a fixed common due date d for all jobs J_j $(j = 1, 2, \ldots, n)$ $(1|d|\sum g_j(E/T))$
2. problems having an individual due date d_j for each J_j $(j = 1, 2, \ldots, n)$ $(1|d_j|\sum g_j(E/T))$
3. problems having the common due date d as a variable for all J_j $(j = 1, 2, \ldots, n)$ (optimal due date assignment) $(1||\sum g_j(E/T))$

B. its criterion that has usually been the minimization of total penalty cost as objective function, where the penalties can be measured in different ways:

1. total unweighted earliness and tardiness (sum of absolute deviations of the job completion times from due date) $(g_j(E/T) = (E_j + T_j))$
2. total weighted earliness and tardiness (weighted sum of absolute deviations of the job completion times from due date) $(g_j(E/T) = (w_j E_j + v_j T_j))$

Apart from these classifications, it is possible to talk about the objective function which is a function of the squared earliness and tardiness penalties. Such a problem, $1|d|\sum(E_j^2+T_j^2)$, has been studied by Bagchi, Sullivan and Chang (1987). They define this problem as unrestricted if an increase in the due date does not result in any further decrease in the objective function value. That is, the due date does not constrain the minimization of mean squared deviation in any way. They have presented a general purpose branching procedure to solve this problem.

Later, De *et al.* (1989) have demonstrated that for the restricted version of this problem, the branching procedure proposed by Bagchi, Sullivan and Chang (1987) does not always produce the optimal schedule. They have suggested an alternative approach that ensures optimality under all circumstances. They have also analyzed the derivation of bounds which are very useful in determining whether the problem is restricted or unrestricted. Furthermore, they have presented a solution procedure for the restricted problem.

Another criterion that has been studied in the literature is the minimization of maximum penalty of earliness and tardiness with different due dates. Sidney (1977) has considered this problem with penalties that are monotonically nondecreasing continuous functions of earliness (measured, in this case, as the deviation of the starting time from a given target starting time) and tardiness. Under restrictive assumptions on the starting times and the due dates, an $\mathcal{O}\ (n^2)$ algorithm for minimizing the maximum penalty has been presented in the paper. For the same problem an $\mathcal{O}\ (n \log n)$ algorithm has been later given by Lakshminarayan *et al.* (1978).

Baker and Scudder (1990) review the literature on the total earliness and tardiness scheduling problems providing a framework to show how results have been generalized starting with a basic model of a single machine common due date total unweighted earliness and tardiness problem. They also investigate more general models which are obtained by adding some features to the basic model such as parallel machines, complex optimality criteria and distinct due dates. Apart from this review, there has been a number of review studies about total earliness and tardiness problems with different emphases.

Sen and Gupta (1984) survey scheduling models involving due dates. After classifying the scheduling problems according to their optimality criteria, they provide the theoretical developments and computational experiences within each category. The optimality criteria considered in this review include the minimization of a criterion related to the flow time of jobs, the set-up cost of machines, a criterion related to job lateness or tardiness, in-process inventories, and a combination of two or more of the above criteria.

Cheng and Gupta (1989) survey models in which the due dates are decision variables for both static and dynamic job shop situations. They analyze the static job shop for the case where the due date is constrained to be greater than or equal to the total processing times. For these type problems, the optimal due date and the optimal sequence are to be determined when the method of assigning due dates is specified. They discuss the identification of the most desirable due date assignment method for the dynamic job shops together with the literature dealing with determination of optimal due dates.

2.3 Mathematical Formulation of the Problem, $1|d_j|\Sigma(w_j E_j + v_j T_j)$

This problem is the most general form of the total earliness and tardiness problems since it includes distinct due dates and weights for every job. This problem can be reduced to all other problems by defining the due dates and the weights appropriately.

NLTETP:

$$\text{Min} \quad \sum_{j=1}^{n}(w_j E_j + v_j T_j)$$

s.t.

$$\begin{aligned}
C_j + E_j - T_j &= d_j & \forall j && (2.1)\\
\sum_{j=1}^{n} C_j x_{ij} - \sum_{j=1}^{n} C_j x_{(i-1)j} - \sum_{j=1}^{n} p_j x_{ij} &\geq 0 & \forall i && (2.2)\\
\sum_{j=1}^{n} x_{ij} &= 1 & \forall i && (2.3)\\
\sum_{i=1}^{n} x_{ij} &= 1 & \forall j && (2.4)\\
x_{ij} &\in \{0,1\} & \forall i,j && (2.5)\\
C_j, E_j, T_j &\geq 0 & \forall j && (2.6)
\end{aligned}$$

where x_{ij} is defined as

$$x_{ij} = \begin{cases} 1 & \text{if } J_j \text{ is assigned to position } i \\ 0 & \text{otherwise} \end{cases}$$

In this model, the objective function is to minimize the weighted total earliness and tardiness of all jobs. Constraint 2.1 describes the earliness or tardiness of each job. From this constraint a job can only be either early or tardy, if it is not an on-time job. Constraint 2.2 prevents jobs to be overlapped. While constraint 2.3 denotes that exactly one job can be processed at every position, constraint 2.4 denotes that each job can be processed exactly at one position, that is no preemption is allowed. It can be seen that the model is nonlinear due to the constraint 2.2. Due to this nonlinearity, the computational difficulty of total earliness and tardiness problems increases. Fry, Leong and Rakes (1987) have provided another formulation for $1|d_j|\sum(w_j\,E_j + v_j\,T_j)$ problem. Since, their formulation does not contain any nonlinearities, it is easier to handle. Their model, called MITETP, is provided below:

MITETP:

$$\text{Min} \quad \sum_{i=1}^{n}(w_i\,E_i + v_i\,T_i)$$

s.t.

$$\begin{aligned}
\sum_{k=1}^{i} Y_k + \sum_{k=1}^{i}\sum_{j=1}^{n} p_j\,x_{kj} + E_i - T_i &= \sum_{j=1}^{n} d_j\,x_{ij} && \forall i && (2.7)\\
\sum_{j=1}^{n} x_{ij} &= 1 && \forall i && (2.8)\\
\sum_{i=1}^{n} x_{ij} &= 1 && \forall j && (2.9)\\
x_{ij} &\in \{0,1\} && \forall i,j && (2.10)\\
E_i, T_i, Y_i &\geq 0 && \forall i && (2.11)
\end{aligned}$$

where Y_i denotes the idle time to be inserted before the job at the i-th position. Hence, constraint 2.7 describes both the position of a job in the schedule, and its condition, that is being early, tardy, or on-time. Constraints 2.8, 2.9 and 2.10 are same as constraints 2.3, 2.4 and 2.5, respectively.

These models are important because the mathematical formulation of the problems is used in finding the optimal solution in our study. Hence, computationally tractable and efficient models are essential. Although the later model is a mixed-integer formulation and it is not efficient for large n values, it is notable in the sense that it overcomes the difficulties regarding the non-linear constraints of the former model. Furthermore, the former model is important because this type of formulation can easily be used in finding the optimal schedule when a sequence is given. Because, once a sequence is determined, we can eliminate all the decision variables of x_{ij}. In other words, since the technical constraints (or the precedence relationships) are all given, the NLTETP model turns out to be a linear programming model. In a formal way, we can state the linear programming model as follows:

LPTETP(σ):

$$\text{Min} \quad \sum_{i=1}^{n}(w_{\sigma(i)}\, E_{\sigma(i)} + v_{\sigma(i)}\, T_{\sigma(i)})$$

s.t.

$$\begin{aligned}
C_{\sigma(i)} + E_{\sigma(i)} - T_{\sigma(i)} &= d_{\sigma(i)} & \forall i && (2.12)\\
C_{\sigma(i)} - C_{\sigma(i-1)} - p_{\sigma(i)} &\geq 0 & \forall i && (2.13)\\
E_{\sigma(i)}, T_{\sigma(i)}, C_{\sigma(i)} &\geq 0 & \forall i && (2.14)
\end{aligned}$$

Note that, $C_{\sigma(i)}$ denotes the completion time where $C_{\sigma(0)} = 0$ and $p_{\sigma(i)}$ denotes the processing time of the i-th job in the sequence σ, respectively. In this case, since a sequence is given, disjunctive constraints 2.3 and 2.4 of the NLTETP model are not necessary in LPTETP model. In this linear programming formulation, we have $3\,n$ variables and $2\,n$ constraints. Although this model can be solved easily, there exists more efficient algorithms in the literature for finding the optimal schedule when a sequence is given (see Section 2.6.4).

2.4 Special Cases

In this section, we present some more definitions which are used throughout the rest of this study. Some of these definitions are also required for stating the special cases for the problem.

Definition 2.7 *In a Weighted Shortest Processing Time* (WSPT) *sequence jobs are ordered according to non-decreasing ratios of processing times to the weight of tardiness, i.e.,*

$$p_1/v_1 \leq p_2/v_2 \leq \ldots \leq p_n/v_n.$$

This sequence is called as Shortest Processing Time (SPT) *sequence if the weight of tardiness is equal to one for all jobs, that is if:*
$v_j = 1 \quad \forall\ j = 1, 2, \ldots, n.$

Definition 2.8 *In a Weighted Longest Processing Time* (WLPT) *sequence jobs are ordered according to non-increasing ratios of processing times to the weight of earliness, i.e.,*

$$p_1/w_1 \geq p_2/w_2 \geq \ldots \geq p_n/w_n.$$

This sequence is called as Longest Processing Time (LPT) *sequence if the weight of earliness is equal to one for all jobs, that is if:*
$w_j = 1 \quad \forall\ j = 1, 2, \ldots, n.$

For total earliness and tardiness problems there exist two special cases which are true for all objective functions and restrictions about due dates (Ow and Morton, 1989).

1. If WSPT sequence results in a schedule with no early jobs, then this is optimal schedule.

2. If WLPT sequence results in a schedule with no tardy jobs, then this is optimal schedule.

Apart from these special cases, since the due dates are involved in the optimality criterion of the earliness and tardiness scheduling problems, it seems natural to have some sequencing rules that incorporates the information of due dates. One of such sequencing rules is the following:

Definition 2.9 *In an Earliest Due Date* (EDD) *sequence jobs are ordered according to non-decreasing values of the due dates, i.e.,*

$$d_1 \leq d_2 \leq \ldots \leq d_n.$$

A second sequencing rule that uses due date information is to consider target starting times (or equivalently the *slack times*) of the jobs as follows:

Definition 2.10 *In an Earliest Starting Time* (EST) *sequence jobs are ordered according to non-decreasing values of the target starting times, i.e.,*

$$a_1 \leq a_2 \leq \ldots \leq a_n.$$

Informally, the slack time of a job is the amount of time remaining before this job must be started if it is to be completed on time. This quantity is important because it shows the urgency of the jobs: smaller the slack time of a job, higher the chance of being tardy for that job.

2.5 Well Solvable Cases

2.5.1 $1|d|\sum(w_j E_j + v_j T_j)$

This problem is considered for two special cases according to the different weight assignments. In the first one all early jobs have the same weight w and all tardy jobs have the same weight v. In the second case, each job J_j has a weight w_j which is independent of whether the job is early or tardy.

2.5.1.1 $1|d \geq P|\sum(w E_j + v T_j)$

The problem $1|d|\sum(w E_j + v T_j)$ is referred as the *weighted sum of absolute deviation* (WSAD) problem in the literature. This problem has been discussed firstly by Bagchi, Chang and Sullivan (1987). The problem $1|d \geq P|\sum(w\,E_j + v\,T_j)$ is called as the unrestricted version of $1|d|\sum(w\,E_j + v\,T_j)$ problem since the due date is loose. The other case in which due date is tight is called as the restricted version. This distinction is important because while the unrestricted version can be solved in polynomial time, the restricted version has been shown to be $\mathcal{NP}$-hard. To be more precise, $1|d|\sum(E_j + T_j)$ problem is defined as unrestricted if $d \geq P$ and restricted if $d < P$, where P is the total processing time of the jobs in a schedule, i.e.,

$$P = \sum_{j=1}^{n} p_j.$$

The difference between two problems can be described qualitatively and in a simplified form. When the problem is unrestricted, since the due date is loose, there is more flexibility to construct the schedule such that d is in the middle of the schedule. In the restricted problem, however, this may not be possible since no job can start before time zero.

Bagchi, Chang and Sullivan (1987) have stated some propositions for the unrestricted version of the problem. These propositions restrict the search for an optimal solution to a limited set of schedules.

Proposition 2.1 *There is no inserted idle time between any jobs in the optimal schedule (If job J_j immediately follows job J_k in the schedule, then $C_j = C_k + p_j$).*

Proof: Assume the contrary. It is easy to see that the value of the objective function can be improved by eliminating the idle time between jobs. ■

Definition 2.11 *A schedule is called V-shaped if the jobs completed on or before the common due date are in the* WLPT *order and the jobs completed after the common due date are in the* WSPT *order.*

Proposition 2.2 *The optimal schedule is V-shaped.*

Proof: The proof follows from a simple job-interchange argument. ∎

Proposition 2.3 *One job completes precisely at the common due date ($C_j = d$ for some job J_j).*

Proof: The proof follows from a shifting argument of the schedule. ∎

Proposition 2.4 *For $d \leq p_1$, the* WSPT *sequence is optimal. Furthermore, if $w < v$, then the* WSPT *sequence is optimal for $d \leq (p_1 + p_2)/2$.*

Proof: For $d \leq p_1$, Proposition 2.4 is a well-known result. For $w < v$ and $d \leq (p_1 + p_2)/2$, the proposition can be established by job-interchange argument. ∎

Proposition 2.5 *In an optimal schedule the longest job is processed first.*

Proof: Consider a schedule of σ that satisfies Proposition 2.2 but violates Proposition 2.5. In such a schedule the longest job must be the last job in sequence. It is easily shown that the interchange of the first and the last jobs in σ improves the schedule. ∎

Proposition 2.6 *For an optimal schedule of the form $(\mathcal{E}')(\mathcal{T})$ one can write*

$$|\mathcal{E}'| = \begin{cases} \lceil n(v/\nu) \rceil & \text{for } n(v/\nu) \text{ non-integer} \\ n(v/\nu) \ \text{ or } \ n(v/\nu) + 1 & \text{for } n(v/\nu) \text{ integer} \end{cases}$$

where $\nu = w + v$.

Proof: Similar to the proof of Proposition 2.7 in Section 2.5.2. ∎

For this problem an $\mathcal{O}$ $(n \log n)$ algorithm is offered by Bagchi, Chang and Sullivan (1987) which is an alternative algorithm to the one given by Panwalkar *et al.* (1982). The algorithm of Bagchi, Chang and Sullivan (1987) yields a unique sequence with optimal due date equal to d^* where d^* is the minimum due date for the optimal objective function. Jobs are scheduled by the algorithm one at a time, starting with the largest job and ending with the shortest job.

Panwalkar *et al.* (1982) have considered the problem of finding the best common due date for all jobs to minimize the total penalty which is the summation of weighted earliness and tardiness cost together with the due date assignment cost. A two-phased algorithm is given for solving this problem. In the first phase, the number of tardy jobs is found. Positional penalties, the optimal sequence and the optimal due date are found in the second phase. The algorithm subsumes the algorithm given by Kanet (1981) for minimizing sum of absolute deviation of completion times about a common due date as a special case.

2.5.1.2 $1|d|\sum w_j\,(E_j + T_j)$

Hall and Posner (1991) have shown that Propositions 2.1, 2.2 and 2.3 hold for the problem $1|d|\sum w_j\,(E_j + T_j)$ and presented four special cases for which the problem $1|d|\sum w_j\,(E_j + T_j)$ is polynomially solvable. Hoogeveen and van de Velde (1992) has given one more special case for the problem $1|d|\sum w_j\,(E_j + T_j)$ which is polynomially solvable:

Weights and Processing Times are Equal In this special case, $w_j = p_j, \quad \forall\ j = 1, 2, \ldots, n$. The algorithm proposed for this problem by Hall and Posner (1991) is in $\mathcal{O}(n)$ and takes jobs in non-increasing order of their processing times and schedules them to finish early, until the total processing time of jobs scheduled is at least half of the overall total processing time. All other jobs are scheduled late. Since $w_j/p_j = 1, \quad \forall\ j = 1, 2, \ldots, n$, the order of the jobs in $\mathcal{E}'$ or $\mathcal{T}$ does not affect the objective function value. This result has also been obtained independently by Hoogeveen and van de Velde (1992).

Jobs Have Unit Processing Times For this class of problems, it is assumed that $p_j = 1, \quad \forall\ j = 1, 2, \ldots, n$. From Propositions 2.2 and 2.3, the completion times of the jobs are $d - \lceil n/2 \rceil + 1, d - \lceil n/2 \rceil + 2, \ldots, d - \lceil n/2 \rceil + n$. With some simple algebraic manipulations, it is obvious that an optimal schedule σ^* has an objective function value of

$$z(\sigma^*) = \sum_{j=1}^{n} \lfloor j/2 \rfloor\, w_j.$$

No Job is Late For this class of problems, Hall and Posner (1991) have given sufficient conditions for all jobs to be in $\mathcal{E}'$.

Theorem 2.1 [Hall and Posner, 1991] *If* $p_k \geq 2\sum_{i=1}^{k-1} p_i$ *and* $w_k \geq 2\sum_{i=k+1}^{n} w_i$ $\forall\ k = 1, 2, \ldots, n$, *then the schedule of* $\sigma = (n, n-1, \cdots, 2, 1)$, *where job* J_1 *is the on-time job, is optimal.*

Proof: Suppose that for some job $J_k \in \mathcal{T}$. Let Δ denote the change in cost from inserting job J_k in its appropriate position in $\mathcal{E}'$. Then

$$\begin{aligned} \Delta &\leq \left(\sum_{i=k+1}^{n} w_i\right) p_k + w_k \sum_{i=1}^{k-1} p_i - w_k p_k \\ &\leq w_k p_k/2 + w_k p_k/2 - w_k p_k = 0. \end{aligned}$$

If $t > 1$, the same operation can be repeated until $\mathcal{T}$ is empty. ■

First Job is Large For this class of problems, Hall and Posner (1991) have shown that if $p_1 \geq \sum_{i=2}^{n} p_i$, that is if there exists one very large and important job that dominates processing, then the schedule of $\sigma = (1, 2, \cdots, n-1, n)$, where job J_1 is the on-time job, is optimal.

Identical Jobs Hoogeveen and van de Velde (1992) have shown that if the jobs are identical, that is, if $p_j = p, \quad \forall\ j = 1, 2, \ldots, n$, then an optimal schedule can be obtained by applying the matching procedure of Emmons (1987).

2.5.2 $1|d \geq \delta|\sum(E_j + T_j)$

For the problem $1|d|\sum(E_j + T_j)$ again restricted and unrestricted versions occur according to the constraints on the common due date d. This problem is called as the unrestricted version of $1|d|\sum(E_j + T_j)$ problem since the due date is loose. Similar to the weighted case, the problem $1|d \geq \delta|\sum(E_j + T_j)$ is polynomially solvable whereas the restricted version is $\mathcal{NP}$-hard. To be more precise, $1|d|\sum(E_j + T_j)$ problem is defined as unrestricted if $d \geq \delta$, where δ is defined as

$$\delta = \begin{cases} p_1 + p_3 + \ldots + p_n & \text{if n is odd} \\ p_2 + p_4 + \ldots + p_n & \text{if n is even} \end{cases}$$

Otherwise, i.e. if $d < \delta$, the problem is defined as restricted. This special case has been studied by Kanet (1981), Hall (1986), and Bagchi *et al.* (1986) under different restrictions on the common due date.

Kanet (1981) has addressed the unrestricted problem and presented an $\mathcal{O}\ (n^2)$ algorithm for minimizing the total cost when costs increase linearly as a job's completion time. More specifically, the objective function is to minimize the sum of absolute lateness for the unrestricted problem. Also, the restriction on the common due date is looser than the one given above. The problem is defined as unrestricted if $d \geq P$.

Hall (1986) has also examined the problem analyzed by Kanet (1981). Considering an equivalent problem, the proof of optimality of Kanet's algorithm is discussed and it is shown that the conditions are not necessary. A new algorithm which finds alternative optimal solutions for the problem is also stated.

Bagchi *et al.* (1986) have also considered the problem $1|d|\sum(E_j + T_j)$ but with different restrictions on common due date d. An $\mathcal{O}\ (n \log n)$ algorithm is suggested for the problem under the restriction that $d \geq \delta$. It is proposed that if $d \geq \theta$, where

$$\theta = p_n + \begin{cases} p_2 + p_4 + \ldots + p_{n-1} & \text{if n is odd} \\ p_1 + p_3 + \ldots + p_{n-1} & \text{if n is even} \end{cases}$$

the algorithm produces multiple optima including Kanet's schedule. The number of optimal schedules, assuming all p_j are different, is $2^{n/2}$ if n is even, and $2^{(n-1)/2}$ if n is odd. If $\delta \leq d < \theta$ then the algorithm yields at least one optimal schedule. Since the bound for this problem is determined by the ordering step of complexity $\mathcal{O}$ $(n \log n)$ and any optimal algorithm for this problem would have to perform this step, the order of computation for this problem cannot be improved any further unless the order of computation for sorting problem is improved.

For the problem $1|d \geq \delta|\sum(E_j + T_j)$, the properties that the optimal solution should possess can be given by Propositions 2.1, 2.2 and 2.3. Proposition 2.1 implies that once a sequence of jobs and a starting time for that sequence are given, it is easy to determine the schedule. So, it is enough to search $n!$ different sequences for an optimum. Proposition 2.2 implies that once the membership in two sets is known, the sequence of the jobs within each set can be determined immediately. This limits the search for an optimum to 2^n sequences instead of all $n!$ sequences. Although the optimal job sequence is determined, without knowing the starting time, there exists an infinite number of schedules to evaluate. Proposition 2.3 says that there exists two sets of jobs, an early set (which includes one job precisely on-time) and a tardy set. By using Proposition 2.3, once the membership in the early set is known, the starting time for that sequence can easily be determined. There are thus 2^n schedules to evaluate in the search for an optimum.

Proposition 2.7 *In an optimal schedule, the $|\mathcal{E}'|$-th job in sequence completes at time d, where $|\mathcal{E}'|$ is as follows:*

$$|\mathcal{E}'| = \begin{cases} \lceil n/2 \rceil & \textit{if } n \textit{ is odd} \\ n/2 \quad \textit{or} \quad (n/2)+1 & \textit{if } n \textit{ is even} \end{cases}$$

Proof: Considering the Propositions 2.1, 2.2 and 2.3, the value of the objective function for a schedule can be determined. Let $\mathcal{E}'i$ denote the index of the i-th job in $\mathcal{E}'$ and let $\mathcal{T}i$ denote the index of the i-th job in $\mathcal{T}$. The penalty due to the deviation of completion time from its due date for job $\mathcal{E}'i$ is the sum of the processing times of all jobs in $\mathcal{E}'$ that complete later. With some algebraic manipulation, the total penalty for the jobs in $\mathcal{E}'$ can be written

$$C_{\mathcal{E}'} = 1p_{\mathcal{E}'1} + 2p_{\mathcal{E}'2} + \cdots + (|\mathcal{E}'| - 1)p_{\mathcal{E}'(|\mathcal{E}'|-1)} + |\mathcal{E}'|\, p_{\mathcal{E}'|\mathcal{E}'|}. \tag{2.15}$$

Similarly, the penalty for job $\mathcal{T}i$ is the sum of the processing times of all jobs in $\mathcal{T}$ that start before plus the processing time of job $\mathcal{T}i$. With some algebraic manipulation, the total penalty for the jobs in $\mathcal{T}$ can be written

$$C_{\mathcal{T}} = (|\mathcal{T}| - 1)p_{\mathcal{T}1} + (|\mathcal{T}| - 2)p_{\mathcal{T}2} + \cdots + 1p_{\mathcal{T}(|\mathcal{T}|-1)} + 0p_{\mathcal{T}|\mathcal{T}|}. \tag{2.16}$$

The objective function is the sum of $C_{\mathcal{E}'}$ and $C_{\mathcal{T}}$ and the processing times are given. If $|\mathcal{T}|$ and $|\mathcal{E}'|$ are known, this sum would be minimized by matching the smallest coefficient in the sum with the largest processing time, the next smallest coefficient with the next largest processing time, and so on, with ties broken arbitrarily. Hence, if n is even, then $|\mathcal{E}'| = |\mathcal{T}|$, and if n is odd, then $|\mathcal{T}| = |\mathcal{E}'| + 1$. ■

2.5.3 $1||\Sigma g_j(E/T)$

Optimal due date assignment problem has been examined by Seidmann *et al.* (1981), Panwalkar *et al.* (1982), Quaddus (1987), Cheng (1987), Bector *et al.* (1988), Cheng (1988a), Cheng (1988b), Baker and Scudder (1989) and Cheng (1990a) with the objective of minimizing the sum of absolute deviation of the job completion times from the optimal common due date.

Seidmann *et al.* (1981) have considered a total aggregate penalty function to be minimized to find the optimal due date for each job and the corresponding optimal sequence. Their penalty function is based on the unweighted per unit lead time penalty dependent upon the specific due date assigned to each individual job, together with the earliness, and tardiness cost of the job represented by P_1, P_2, and P_3, respectively. It is shown that if $P_1 \leq P_3$, then $d_j = p_j, \quad \forall \ j = 1, 2, \ldots, n$, otherwise $d_j = \min\{A, \sum_{1 \leq j \leq i} p_j\}, \quad j = 1, 2, \ldots, n$, where A represents the lead time that customers consider to be reasonable and expected. It is also proved that the SPT sequence will be optimal for this type of problems.

Panwalkar *et al.* (1982) have also considered the same type of objective function with Seidmann *et al.* (1981) with a slight variation. In this case, P_1 is the per unit time cost of due date. An algorithm is provided to determine an optimal sequence and the corresponding optimal common due date. It is proved that for any specified sequence σ, there exists an optimal common due date equal to $C_{\sigma(i)}$, where

$$i = \lceil n\,(P_3 - P_1)/(P_2 + P_3) \rceil.$$

Quaddus (1987) has presented a linear programming analysis for assigning an optimal due date to n independent jobs. The criterion for this problem is the minimization of total unweighted earliness and tardiness penalties together with the penalty assessed on due date allowance.

Cheng (1987) has considered the problem of finding the optimal common due date and the optimal job sequence to minimize the weighted average of missed due dates. He has proposed an exponential algorithm to search the optimal solution which is efficient for small to medium values of n.

Bector *et al.* (1988) have considered to find the optimal common due date and a corresponding optimal sequence such that the total unweighted earliness and tardiness penalties are minimized. A generalized linear goal program is considered to prove some basic results. Then, an algorithm is developed using these results and the idea of sensitivity analysis in linear programming which determines the optimal common due date and the corresponding optimal sequence.

Cheng (1988a) has proved the same result of Panwalkar *et al.* (1982), using the duality property of linear programming for this type of performance measure. Cheng (1988b) has considered the problem of determining optimal common due date and optimal sequence of jobs for minimizing unweighted total earliness and tardiness penalties. The main difference of this work is that a deviation parameter is considered in the due date assignment problem which attempts to model the real life situation of no penalty for jobs marginally missing the common due date.

Baker and Scudder (1989) have given an alternative proof of the result of Quaddus (1987), without relying on duality theory, and have shown how Quaddus' examples fall short of optimizing the total penalty. This is due to neglecting the sequencing aspect of the problem while considering the job dependent penalties, thereby generalizing models addressed by other authors.

Cheng (1990a) has provided a new algorithm for the same problem of Cheng (1987) which is significantly more efficient than the former one due to a partial search algorithm incorporated into algorithm of Cheng (1987).

2.6 $\mathcal{NP}$-hard Problems

2.6.1 $1|d < \delta|\sum(w_j E_j + v_j T_j)$

Hall and Posner (1991) have shown that this problem with $w_j = v_j$, $\forall\ j = 1, 2, \ldots, n$, is $\mathcal{NP}$-hard in the ordinary sense by providing a reduction from the EVEN-ODD PARTITION problem (see Section 2.6.4) which is already known to be $\mathcal{NP}$-complete in the ordinary sense (Garey *et al.* 1988). After describing optimality conditions for the problem, a computationally efficient dynamic programming algorithm is developed. A fully polynomial approximation scheme is given when the weights are bounded by a polynomial function of the number of jobs.

Hoogeveen and van de Velde (1992) have also provided an alternative proof of showing that $1|d < \delta|\sum(w_j E_j + v_j T_j)$ is $\mathcal{NP}$-hard in the ordinary sense even if all job weights are equal (that is $w_j = v_j = 1$) by a reduction from the EVEN-ODD PARTITION problem. They have presented a pseudo-polynomial algorithm that requires $\mathcal{O}\ (n^2 d)$ time and $\mathcal{O}\ (nd)$ space.

For $1|d < \delta|\sum(w_j E_j + v_j T_j)$ with $w_j = w$ and $v_j = v, \quad \forall\ j = 1, 2, \ldots, n$ Propositions 2.1 and 2.2 hold but Proposition 2.3 does not. For solving this problem, Bagchi, Chang and Sullivan (1987) have proposed a branching procedure based on Proposition 2.2 and Proposition 2.4.

2.6.2 $1|d < \delta|\sum(E_j + T_j)$

Hall *et al.* (1991) have shown that the problem $1|d < \delta|\sum(E_j + T_j)$ is $\mathcal{NP}$-hard by transforming the EVEN-ODD PARTITION problem to the recognition question of the problem $1|d < \delta|\sum(E_j + T_j)$. It is demonstrated that the problem $1|d < \delta|\sum(E_j + T_j)$ is not $\mathcal{NP}$-hard in the strong sense by describing a pseudo-polynomial time algorithm.

Bagchi *et al.* (1986) have developed a branching procedure for the problem $1|d < \delta|\sum(E_j + T_j)$. For $d = \delta$, there exists an optimal schedule in which jobs are processed continuously from $t = 0$ to $t = P$. When $d < \delta$, again in an optimal schedule jobs are processed in the interval of $[0, P]$. However there exists now no guarantee that an optimal schedule will contain no job J_j with $S_j < d$ and $C_j > d$. In the proposed procedure to determine an optimal schedule for this problem which is based on implicit enumeration, some dominance properties are utilized to curtail the search of all feasible V-shaped schedules.

Bagchi *et al.* (1986) have reported that the assumption of processing the jobs in the interval $[0, P]$, i.e. $C_{\sigma(n)} = P$, is unnecessary when $d \leq \theta$. However, it becomes a constraint when $d > \theta$.

For the restricted version of the problem, Propositions 2.1 and 2.2 still hold but

Propositions 2.3 and 2.7 do not. As a special case, it should be noted that if $d = 0$, then the problem is solved optimally by the SPT rule.

Raghavachari (1986) has proved the V-shape property of optimal schedule of jobs about a common due date. It is shown that for the problem of minimizing the sum of absolute deviation of the job completion times about a common due date, the optimal sequence is always V-shaped and this property is not affected by the restrictions on the due date.

Sundararaghavan and Ahmed (1984) have investigated the problem $1|d < \delta|\sum(E_j + T_j)$ in which the job in the first position in the schedule is assumed to start at time zero. A heuristic is proposed for this problem and its performance is reported on several randomly generated problems. The algorithm finds the optimal solution in most of the cases and it usually fails only by a small margin. Szwarc (1989) has also considered this problem and developed a branch-and-bound algorithm which incorporates a lower bound and utilizes the heuristic procedure of Sundararaghavan and Ahmed (1984) for finding the initial upper bound.

2.6.3 $1|d_j|\sum(w_j E_j + v_j T_j)$

$1|d_j|\sum(w_j E_j + v_j T_j)$ is $\mathcal{NP}$-hard, since its special case $1|d_j|\sum(E_j + T_j)$ is so (see Section 2.6.4). This problem has been studied by Tahboub (1987), Fry, Armstrong and Blackstone (1987), Ow and Morton (1988), Ow and Morton (1989), Yano and Kim (1989), Azizoğlu *et al.* (1991) and Hoogeveen (1992).

Tahboub (1987) has investigated the use of a surrogate problem in solving this problem. This approach exploits the relationship of the problem to a transportation problem that may be used to decompose the original problem into subproblems for which optimal solutions may be easily obtained and then re-composed to determine the solution of the problem.

Fry, Armstrong and Blackstone (1987) have presented a heuristic solution for the problem since the problem has a non-transitive penalty function and hence requires an enumeration procedure to guarantee a global optimum. In this study, insertion of idle time to obtain an optimal delay schedule is performed by solving a linear program with the objective of minimizing weighted sum of absolute deviations. The second aspect of the heuristic is to sequence the jobs. The best results are obtained by using a sequencing rule such as EDD to sequence the jobs and then by applying adjacent pairwise interchange.

Ow and Morton (1988) have considered this problem as one of the problems in their experimental study for a new search method they have developed. Later, Ow and Morton (1989) have analyzed this problem individually proposing two dispatch priority rules together with the new search method.

Yano and Kim (1989) have studied this problem and have presented a dynamic programming algorithm to find the optimal schedule given a fixed sequence for the jobs. They have developed optimal and heuristic procedures when the weights are proportional to the processing times of the jobs. They have also derived dominance criteria which are used both in the optimal branch-and-bound procedure to reduce the number of sequences that must be considered, and in the heuristic procedure to construct an initial sequence and to evaluate potential improvements. This heuristic procedure combines a simple sorting procedure and a simple pairwise interchange procedure. Both the optimal and the heuristic procedures are variations of the procedures presented in Sections 2.6.4.2 and 2.6.4.3.

Azizoğlu *et al.* (1991) have examined the problem where jobs have the same earliness and tardiness penalties which are penalized at different rates. They have assumed that no inserted idle time exists in the schedule. For this problem, they have presented a branch-and-bound algorithm together with powerful lower and upper bounds.

Hoogeveen (1992) has considered the problem $1|d_j|\sum((w_j+v_j)\,E_j+v_j\,C_j)$ where $w_j = w$ and $v_j = v, \quad \forall \quad j = 1,2,\ldots,n$. This problem is equivalent to the problem $1|d_j|\sum(w_j E_j + v_j T_j)$ since $C_j = T_j - E_j + d_j \quad \forall\ j = 1, 2, \ldots, n$. He proposed a branch-and-bound algorithm for this problem by determining some lower bounds and upper bounds (see Section 2.6.4.2).

2.6.4 $1|d_j|\sum(E_j + T_j)$

Garey *et al.* (1988) have shown that the problem $1|d_j|\sum(E_j + T_j)$ is $\mathcal{NP}$-hard by a reduction from the EVEN-ODD PARTITION problem.

EVEN-ODD PARTITION:

INSTANCE: An integer n, a finite set $\mathcal{A}$ of $2n$ elements, a "size" $a_j \in \mathcal{Z}^+$ for each $A_j \in \mathcal{A}$, such that $a_j < a_{j+1}$ for each $A_j \in \mathcal{A}$, $1 \leq j < 2n$.

QUESTION: Does there exist a partition of $\mathcal{A}$ into subsets $\mathcal{A}_1$ and $\mathcal{A}_2$, such that $\sum_{A_j \in \mathcal{A}_1} a_j = \sum_{A_j \in \mathcal{A}_2} a_j$, and such that for each j, $1 \leq j \leq n$, $\mathcal{A}_1$ and $\mathcal{A}_2$ each contains exactly one of $\{A_{2j-1}, A_{2j}\}$?

TOTAL EARLINESS-TARDINESS:

INSTANCE: Integers y and n, a finite set $\mathcal{J}$ of n independent jobs, a "processing time" $p_j \in \mathcal{Z}^+$ and a "due date" $d_j \in \mathcal{Z}^+$ for each job $J_j \in \mathcal{J}$, $1 \leq j \leq n$.

QUESTION: Does there exist a non-preemptive schedule of these n jobs in the set $\mathcal{J}$ on one machine such that the total earliness and tardiness is no longer than y?

This complexity result indicates that the existence of a polynomially bounded algorithm to determine an optimal solution for the problem is unlikely, so either an implicit enumeration algorithm is required or some heuristic procedures can be used for approximate solutions to the problem. Although there exist a vast amount of literature on the total earliness and tardiness scheduling problems, most of these deal with either the common due date models or the models where due dates are decision variables of the problem. To the best of our knowledge, there exists two branch-and-bound algorithms as optimizing algorithms and two heuristic procedures for $1|d_j|\sum(E_j + T_j)$. Since the problem $1|d_j|\sum(E_j + T_j)$ is more realistic regarding the problems $1|d|\sum(E_j + T_j)$ and $1||\sum(E_j + T_j)$, we concentrate on this problem during this study. For the problem $1|d_j|\sum(E_j+T_j)$, we present a special case of the problem from the literature where we are given a sequence of jobs and it is required to find the optimal schedule for this fixed job ordering in Section 2.6.4.1. In Section 2.6.4.2, we review two branch-and-bound algorithms for the problem $1|d_j|\sum(E_j + T_j)$. Two heuristic procedures are given in Section 2.6.4.3.

2.6.4.1 Scheduling of a Fixed Job Ordering for $1|d_j|\sum(E_j + T_j)$

The words 'sequence' and 'schedule' are different for the problem $1|d_j|\sum(E_j+T_j)$. A sequence defines the order of the jobs without determining the exact place of the jobs in the time horizon. In other words, a sequence only defines the precedence relations between jobs. But a schedule defines the place of the jobs in the time horizon, that is once a schedule is given, the completion times of the jobs are known for sure. For a regular performance measure, a sequence exactly corresponds to the desired schedule. But for a non-regular performance measure such as minimizing the total earliness and tardiness, since insertion of idle time is possible, a sequence does not determine the corresponding schedule. The important point is not to confuse with the 'optimal schedule for a fixed job ordering' and the 'optimal schedule'. Optimal schedule is what we understand from a conventional optimal schedule and it finds the optimal schedule for the given problem instance. In this case, the sequence of the jobs is not our concern and all of the sequences are evaluated in one way or another during the process of finding the optimal schedule. But the optimal schedule for a fixed job ordering considers only the given sequence and finds the optimal timing of the jobs for the given precedence relations. So, in this case, we cannot talk about the optimal schedule for the problem without these precedence relations. Hence, the optimal schedule for a fixed job ordering may not match with the optimal schedule of the problem without any given job ordering. So, for the problem $1|d_j|\sum(E_j + T_j)$, determining the optimal schedule for a fixed job ordering is an additional aspect that has to be considered separately. An algorithm that effectively and efficiently determines the optimal schedule when a sequence is given, is important and worthwhile, since it can be used both in optimizing algorithms such as in a branch-and-bound algorithm, and in heuristic procedures.

Kim and Yano (1987) presented a polynomial algorithm for finding the optimal schedule, when a sequence is given. The procedure is essentially a dynamic programming algorithm. Let $f_{\sigma(i)}(s)$ be minimum tardiness plus earliness of jobs $J_{\sigma(i)}, J_{\sigma(i+1)}, \ldots, J_{\sigma(n)}$, if job $J_{\sigma(i)}$ starts at time s, that is $S_{\sigma(i)} = s$, the recursion equations are:

$$f_{\sigma(i)}(s) = \min_{s\geq 0} \{|s + p_{\sigma(i)} - d_{\sigma(i)}| + \min_{t\geq s+p_{\sigma(i)}} f_{\sigma(i+1)}(t)\}, \quad \forall\ i = 1,2,\ldots,n-1,$$

$$f_{\sigma(n)}(s) = \min_{s\geq 0} |s + p_{\sigma(n)} - d_{\sigma(n)}|$$

where the subscript $\sigma(i)$ refers to the i-th job in the given sequence σ. The internal minimization over t in $f_{\sigma(i)}(s)$ allows idle time between jobs $J_{\sigma(i)}$ and $J_{\sigma(i+1)}$.

In this algorithm, called $OPTSCH|\sigma_{KY}$, the jobs are considered in reverse order of the given sequence. $OPTSCH|\sigma_{KY}$ starts with the last job $J_{\sigma(n)}$ in the given sequence and schedules it to be completed at its due date. Job $J_{\sigma(n-1)}$ can also be scheduled to be completed at its due date unless doing so would cause a conflict with job $J_{\sigma(n)}$. If there is a conflict, the two jobs form a partial sequence with no inserted idle time, which can be scheduled to minimize the corresponding convex function. $OPTSCH|\sigma_{KY}$ proceeds in this fashion, with jobs $J_{\sigma(n-2)}, J_{\sigma(n-3)}, \ldots, J_{\sigma(1)}$, clustering jobs as conflicts occur, until all jobs are scheduled. If the starting time of job J_1 is before time zero, beginning with the initial jobs, partial sequences with no idle time are delayed as little as possible to achieve feasibility by eliminating some or all of the idle time between groups of jobs. The algorithm $OPTSCH|\sigma_{KY}$ has a computational complexity of $\mathcal{O}\ (n^2)$.

Another algorithm for finding optimal schedule when a sequence is given, is presented by Garey *et al.* (1988). In their algorithm, they assume that job $J_{\sigma(i)}$, $\forall\ i = 1, 2, \ldots, n$, must be completed by the time job $J_{\sigma(i+1)}$ is started. Their scheduling algorithm schedules jobs one at a time, shifting previously scheduled jobs earlier when scheduling a new job, if necessary. For this scheduling algorithm, called $OPTSCH|\sigma_{GTW}$, let σ_i be the schedule computed for the first i jobs. $OPTSCH|\sigma_{GTW}$ uses the concept of blocks. Now, assume that there are ℓ blocks, $B_1, B_2, \ldots, B_\ell$ in σ_i. In $OPTSCH|\sigma_{GTW}$, each block B_l is partitioned into two subsets, $Decrease(l)$ and $Increase(l)$, as follows:

$$Decrease(l) = \{J_{\sigma(i)} | J_{\sigma(i)} \in B_l \vee S_{\sigma(i)} > a_{\sigma(i)}\} \quad \text{with} Dec(l) = |Decrease(l)|,$$

$$Increase(l) = \{J_{\sigma(i)} | J_{\sigma(i)} \in B_l \vee S_{\sigma(i)} \leq a_{\sigma(i)}\} \quad \text{with} Inc(l) = |Increase(l)|.$$

We can decrease the penalty of job $J_{\sigma(i)}$ by reducing $S_{\sigma(i)}$ if $J_{\sigma(i)} \in Decrease(l)$, whereas we can increase the penalty of job $J_{\sigma(i)}$ by reducing $S_{\sigma(i)}$ if $J_{\sigma(i)} \in Increase(l)$. Furthermore, define the followings:

$first(l)$: the smallest index of jobs in the block B_l

$last(l)$: the largest index of jobs in the block B_l.

The initial schedule σ_1 simply schedules job $J_{\sigma(1)}$ such that $S_{\sigma(1)} = a_{\sigma(1)}$. In general, given σ_i, job $J_{\sigma(i+1)}$ is scheduled as:

$$S_{\sigma(i+1)} = \begin{cases} a_{\sigma(i+1)} & \text{if } S_{\sigma(i)} + p_{\sigma(i)} \leq a_{\sigma(i+1)} \\ S_{\sigma(i)} + p_{\sigma(i)} & \text{if } S_{\sigma(i)} + p_{\sigma(i)} > a_{\sigma(i+1)} \end{cases}$$

In the first case, job $J_{\sigma(i+1)}$ has no penalty, and σ_i and σ_{i+1} have the same objective function value. In the second case, job $J_{\sigma(i+1)}$ has a positive penalty and both the last block B_ℓ and the corresponding set $Decrease(\ell)$ have gained job $J_{\sigma(i+1)}$ as a member. A key property that $OPTSCH|\sigma_{GTW}$ will maintain is that, for each block B_l in σ_i, either $Dec(l) < Inc(l)$ or $S_{first(l)} = 0$ (in which case $l = 1$). Hence, after scheduling job $J_{\sigma(i+1)}$, we have either $Dec(\ell) \leq Inc(\ell)$ or $S_{first(\ell)} = 0$.

If $S_{first(\ell)} = 0$ or $Dec(\ell) < Inc(\ell)$, we take no further action; the current schedule is σ_{i+1}. On the other hand, if now $Dec(\ell) = Inc(\ell)$ and $S_{first(\ell)} \neq 0$, we can shift the entire block B_ℓ earlier without affecting the objective function value of the schedule. We shift block B_ℓ earlier until one of three things happens:

(i) $S_{first(\ell)}$ becomes zero,

(ii) for some job $J_{\sigma(i)} \in B_\ell$, S_j becomes equal to a_j, or

(iii) $S_{first(\ell)}$ becomes equal to $S_{last(\ell-1)} + p_{last(\ell-1)}$.

The resulting schedule is σ_{i+1}.

$OPTSCH|\sigma_{GTW}$ consists of beginning with the schedule σ_1 and applying the above construction to form schedules $\sigma_2, \sigma_3, \ldots, \sigma_n$. $OPTSCH|\sigma_{GTW}$ can be implemented easily to run in $\mathcal{O}\ (n^2)$ time. But Garey *et al.* (1988) proposed an efficient implementation of $OPTSCH|\sigma_{GTW}$ to run in $\mathcal{O}\ (n \log n)$ time.

2.6.4.2 Optimizing Algorithms

A quick review of the scheduling literature suggests that there is a lack of efficient optimizing algorithms for problem $1|d_j|\sum(E_j + T_j)$. In the following section, we will analyze the algorithms based on branch-and-bound approach.

A Branch-and-Bound Algorithm [Kim and Yano, 1987] As an enumerative algorithm, Kim and Yano (1987) presented a branch-and-bound algorithm which can handle up to 30 jobs. In this branch-and-bound algorithm, they used a lower bound which conceptually says that jobs must be shifted far enough forward or backward in time from their respective due dates to satisfy the constraint of at most one job can be processed on the machine at a given time. The lower bound on the total earliness and tardiness gives the minimal amount of shifting to accomplish this on the assumption that each subset of jobs can be considered separately. In general, shifting in one subset will affect other subsets, so that the given bound may not be achievable.

The following proposition is used for pruning some subproblems in the branch-and-bound algorithm.

Proposition 2.8 [Kim and Yano, 1987] *If there is a constraint such that*

$$\min\{S_{j_1}, S_{j_2}\} + \min\{p_{j_1}, p_{j_2}\} \geq \max\{d_{j_1}, d_{j_2}\},$$

then job J_{j_1} should precede job J_{j_2} when $p_{j_1} < p_{j_2}$, and job J_{j_2} should precede job J_{j_1} when $p_{j_1} > p_{j_2}$.

Proof: Let $ET_{j_1 j_2}$ denote the minimum total earliness and tardiness of jobs J_{j_1} and J_{j_2} when J_{j_1} precedes J_{j_2} and $ET_{j_2 j_1}$ denotes the earliness and tardiness of jobs J_{j_1} and J_{j_2} when J_{j_2} precedes J_{j_1}. Further, let $c_{max} = \max\{C_{j_1}, C_{j_2}\}$. Then

$$ET_{j_1 j_2} = c_{max} + p_{j_1} - d_{j_1} + p_{j_1} + p_{j_2} - d_{j_2},$$

$$ET_{j_2 j_1} = c_{max} + p_{j_2} + p_{j_1} - d_{j_1} + p_{j_2} - d_{j_2},$$

$$ET_{j_1 j_2} - ET_{j_2 j_1} = p_{j_1} - p_{j_2}.$$

Hence, the results follow. ■

Each node of the branching tree is associated with a partial sequence which will be placed at the end of the complete sequence. That is, a node at the k-th level of the tree corresponds to a partial schedule of $\sigma' = \{J_{\sigma'(k)}, J_{\sigma'(k-1)}, \ldots, J_{\sigma'(2)}, J_{\sigma'(1)}\}$, where $J_{\sigma'(i)}$ represents i-th job from the end, and one of the remaining $n-k$ jobs is to be selected for the position $k+1$. Let $\mathcal{J}$ be the set of all jobs, σ' be a partial sequence being placed at the end of the sequence, and $\mathcal{J}_{\sigma'}$ be the set of jobs in σ'. Then the algorithm is as follows:

1. <u>Branching</u>

    - Select a node with the least lower bound in branching tree for branching.

2. <u>Bounding</u>

    - Bound $Bound_1$ for jobs in $\mathcal{J}_{\sigma'}$.
      $Bound_1$ is the optimal solution obtained from $OPTSCH|\sigma_{KY}$ for the sequence σ' under the constraint that the earliest possible start time of the first job in $\mathcal{J}_{\sigma'}$ is $\sum_{J_j \in \mathcal{J} \setminus \mathcal{J}_{\sigma'}} p_j$ instead of 0.
    - Bound $Bound_2$ for jobs in $\mathcal{J} \setminus \mathcal{J}_{\sigma'}$.

    Then the lower bound of the node associated with the partial sequence σ' will be $LB = Bound_1 + Bound_2$.

3. <u>Pruning</u>

    - For pruning, Proposition 2.8 is applied to the partial sequence σ'.

The lower bound used in this branch-and-bound algorithm is not very tight, and furthermore, only one dominance property is used for pruning the partial sequences. The computational results of Kim and Yano (1987) have shown that this branch-and-bound algorithm can solve problems only up to 30 jobs.

A Branch-and-Bound Algorithm [Hoogeveen, 1992] Hoogeveen (1992) has presented a branch-and-bound algorithm for the problem $1|d_j|\sum((w_j + v_j)E_j + v_jC_j)$ where $w_j = w$ and $v_j = v$, $\forall \ j = 1, 2, \ldots, n$. This problem is equivalent to the problem $1|d_j|\sum(E_j + T_j)$ if $w = v = 1$ since $C_j = T_j - E_j + d_j$, $\forall \ j = 1, 2, \ldots, n$. So the proposed branch-and-bound algorithm can be used for the problem $1|d_j|\sum(E_j + T_j)$ by defining the weights appropriately.

The branch-and-bound algorithm of Hoogeveen (1992) can be summarized as follows:

1. Upper Bound

   - Before entering the search tree, determine an *upper bound* (UB) on the optimal solution value. Use the optimal schedule corresponding to the EST sequence as an initial solution, and try to reduce its cost by adjacent pairwise interchange.

2. Branching

   - Adopt a *backward sequencing*. A node at level k of the search tree corresponds to a partial sequence σ' with k jobs fixed in the last k positions with the assumption that the first job in a partial schedule of σ' is not started before time $\sum_{J_j \in \mathcal{J} \setminus \mathcal{J}_{\sigma'}} p_j$.
   - Employ a *depth-first* strategy to explore the tree: at each level, generate the descendant nodes for only one node at a time. At level k, there are $n-k$ descendant nodes: one for each unscheduled job. The completion times for the jobs in σ' are only temporary. Branching from a node that corresponds to σ', add some job J_j leading to the sequence $J_j\sigma'$. Subsequently, determine the associated optimal schedule for $J_j\,\sigma'$, and possibly shift to the right some jobs in σ'. Branch from the nodes in order of non-increasing due dates of the associated jobs.

3. Bounding

   - Use five lower bounds:
     - relax the objective function (not applicable in problem $1|d_j|\sum(E_j+T_j)$)
     - relax the machine capacity (not applicable in problem $1|d_j|\sum(E_j+T_j)$)
     - relax the due dates

        * The common due date problem
        * The common slack time problem
    - relax the processing times
    - Lagrangian relaxation

4. <u>Pruning</u>

    - Discard a node if its associated partial schedule of σ' cannot lead to a complete schedule with cost less than UB; UB denotes the currently best solution value. Let $LB(\mathcal{J} \setminus \mathcal{J}_{\sigma'})$ be some lower bound on the minimal cost of scheduling the jobs in the set $\mathcal{J} \setminus \mathcal{J}_{\sigma'}$. Obviously, discard a node if $z(\sigma') + LB(\mathcal{J} \setminus \mathcal{J}_{\sigma'}) \geq UB$.
    - Use dominance properties to discard or not to discard partial sequence σ'.

The efficiency of this branch-and-bound algorithm was tested on problems with $n = 8, 10, 12, 15, 20$. The computational results have shown that problems up to 10 jobs are easy, but others require considerable computational times.

2.6.4.3 Heuristic Algorithms

A Heuristic Algorithm [Kim and Yano, 1987] Kim and Yano (1987) also gave a heuristic algorithm and stated that in the problems where an optimal solution was found by the branch-and-bound algorithm presented in Section 2.6.4.2, over 80 percent of the solutions from their heuristic algorithm were optimal solutions. But their computational experiment contains only 46 random problems in which the maximum job number is 40. Their heuristic algorithm called as $HEUR_{KY}$ makes use of the following lemma to construct a good sequence, and to compare all pairs of jobs.

Lemma 2.1 [Kim and Yano, 1987] *If there is a conflict between two and only two jobs when the jobs are placed as $C_j = d_j$ for both jobs, and if $p_{j_1} < p_{j_2}$, then the following statements are true.*

Case 1. *If $d_{j_1} > d_{j_2}$, then job J_{j_2} should precede job J_{j_1}.*

Case 2. *If $d_{j_1} < d_{j_2}$, then*

Subcase 2.1. *If $p_{j_1} + (d_{j_2} - d_{j_1}) \geq p_{j_2}$, then job J_{j_1} should precede job J_{j_2}.*

Subcase 2.2. *If $p_{j_1} + (d_{j_2} - d_{j_1}) < p_{j_2}$, let $A = p_{j_2} - p_{j_1} - (d_{j_2} - d_{j_1})$. If $A \leq d_{j_2} - d_{j_1}$, then job J_{j_1} should precede job J_{j_2}, otherwise job J_{j_2} should precede job J_{j_1}.*

Proof:

Case 1. $ET_{j_1 j_2} = p_{j_2} - (d_{j_1} - d_{j_2})$, $ET_{j_2 j_1} = p_{j_1} - (d_{j_1} - d_{j_2})$. Since $p_{j_1} < p_{j_2}$, $ET_{j_2 j_1} < ET_{j_1 j_2}$.

Case 2. $ET_{j_1 j_2} = p_{j_2} - (d_{j_2} - d_{j_1})$, $ET_{j_2 j_1} = p_{j_1} + (d_{j_2} - d_{j_1})$.

Subcase 2.1. Since $p_{j_1} < p_{j_2}$, and $d_{j_1} < d_{j_2}$, $ET_{j_2 j_1} \geq p_{j_2} \geq ET_{j_1 j_2}$.

Subcase 2.2. When $A \leq d_{j_2} - d_{j_1}$,

$$ET_{j_1 j_2} = A + p_{j_1} \leq p_{j_1} + (d_{j_2} - d_{j_1}) = ET_{j_2 j_1}.$$

When $A > d_{j_2} - d_{j_1}$,

$$ET_{j_1 j_2} = A + p_{j_1} \geq p_{j_1} + (d_{j_2} - d_{j_1}) = ET_{j_2 j_1}.$$

■

$HEUR_{KY}$ can be given as follows, where $label_j$ denotes the label of job J_j:

1. Compare jobs J_{j_1} and J_{j_2}, for all possible combinations of two jobs, using the simple rule in Lemma 2.1. If J_{j_1} precedes J_{j_2}, let $label_{j_1} = label_{j_1} - 1$, $label_{j_2} = label_{j_2} + 1$, if J_{j_2} precedes J_{j_1}, let $label_{j_1} = label_{j_1} + 1$, $label_{j_2} = label_{j_2} - 1$. Else, no change in $label_{j_1}, label_{j_2}$.

2. Sort the jobs in non-decreasing order of $label_j$'s. This will be the initial sequence σ.

3. Obtain the optimal timing for the sequence σ resulted from Step 2 using $OPTSCH|\sigma_{KY}$.

4. Check the conditions for dominance in Propositions 2.8 for adjacent pairs of jobs, and change the order if needed. Apply an adjacent pairwise interchange to the sequence.

5. For the sequence resulting from Step 4, obtain optimal timing using $OPTSCH|\sigma_{KY}$. Terminate.

The computational complexity of $HEUR_{KY}$ is $\mathcal{O}(n^2)$. This algorithm is used to test the performance of the heuristic proposed in Section 3.3 and the results are discussed in that section.

A Heuristic Algorithm [Fry *et al.* 1990] Fry *et al.* (1990) also presented a heuristic algorithm to minimize mean absolute lateness which is indeed equivalent to minimize total unweighted earliness and tardiness penalties. This heuristic algorithm, called as $HEUR_{FAR}$, evaluates nine different local optima and chooses the best one as its solution. $HEUR_{FAR}$ utilizes Propositions 3.1, 3.2 and 3.3 which are known in the literature widely. In addition, Fry *et al.* (1990) proposed the following two propositions to be utilized in $HEUR_{FAR}$.

Proposition 2.9 [Fry *et al.* , 1990] *Consider a sequence with adjacent jobs J_{j_1} and J_{j_2} such that job J_{j_1} is tardy regardless of sequence and job J_{j_2} is tardy only if job J_{j_1} precedes job J_{j_2}. If $p_{j_1} \leq p_{j_2}$ and $(d_{j_2} - \min\{S_{j_1}, S_{j_2}\}) \geq (p_{j_2} - p_{j_1})/2$, then job J_{j_1} precedes job J_{j_2} always.*

Proof: The difference between the objective function values between the sequences with job J_{j_1} precedes job J_{j_2} and job J_{j_2} precedes job J_{j_1} is $(d_{j_2} - \min\{S_{j_1}, S_{j_2}\}) \geq (p_{j_2} - p_{j_1})/2$. ■

Proposition 2.10 [Fry *et al.* , 1990] *Consider a sequence with adjacent jobs J_{j_1} and J_{j_2}. If the job sequenced first in the job pair is early and the job sequenced second is tardy, and $d_{j_2} - p_{j_2} \geq d_{j_1} - p_{j_1}$, then job J_{j_1} precedes job J_{j_2} always.*

But Proposition 2.10 is not a correct statement as shown by the following counter example.

Example: Consider two jobs such that $p_{j_1} = 11$, $p_{j_2} = 8$, $d_{j_1} = 20$, and $d_{j_2} = 18$. We assume that jobs J_{j_1} and J_{j_2} are adjacent jobs in any sequence and the interchanging argument does not affect the other jobs in the sequence. Hence, $\min\{S_{j_1}, S_{j_2}\}$ and $\max\{C_{j_1}, C_{j_2}\}$ remain unchanged after the interchange argument. Now, it is obvious that $d_{j_2} - p_{j_2} = 18 - 8 = 10$ which is greater than $d_{j_1} - p_{j_1} = 20 - 11 = 9$. Now consider the case where $\min\{S_{j_1}, S_{j_2}\} = 6$. In this case, whichever the job sequenced first in the job pair will be early and the other job will be tardy as shown in the Figure 2.1.

$$ET_{j_1 j_2} = (20 - 17) + (25 - 18) = 3 + 7 = 10,$$

$$ET_{j_2 j_1} = (18 - 14) + (25 - 20) = 4 + 5 = 9$$

Hence, $ET_{j_1 j_2} - ET_{j_2 j_1} = 10 - 9 = 1 > 0$. So, the sequence where job J_{j_2} precedes job J_{j_1} is better than the sequence where job J_{j_1} precedes job J_{j_2}.

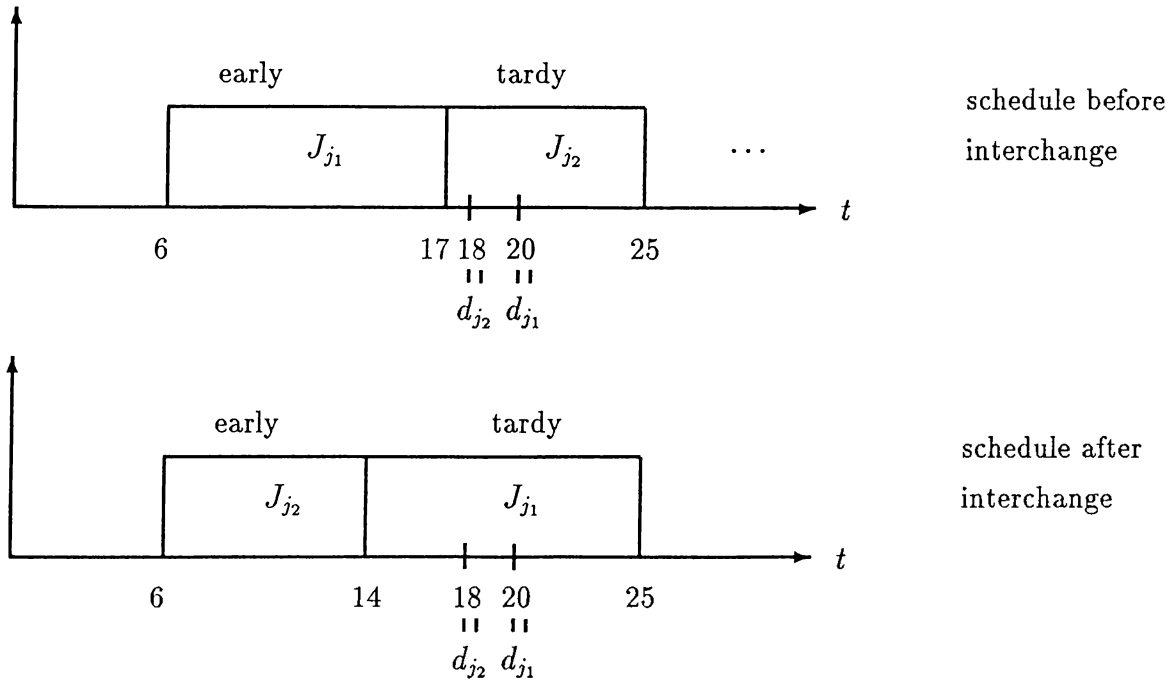


Figure 2.1: Job pair for the counter example

The algorithm $HEUR_{FAR}$ can be given as follows:

1. Initialize system.
2. Set $bestsolution = somelargenumber$.
3. Let $sequencecounter = 0$.
4. Let $sequencecounter = sequencecounter + 1$.
5. If ($sequencecounter = 1$), then let $initialsequence = \text{EDD}sequence$
   Elseif ($sequencecounter = 2$), then let $initialsequence = \text{EST}sequence$
   Otherwise, let $initialsequence = randomsequence$.
6. Let $switchcounter = 0$.
7. Let $switchcounter = switchcounter + 1$.

8. If (*switchcounter* $= 1$), then begin adjacent pairwise interchange (API) at the first position starting with the *initialsequence*. For each pair considered for switching, idle time is inserted using the LPTETP$'$ model.

   Elseif (*switchcounter* $= 2$), then begin API at the last position starting with the *initialsequence*. For each pair considered for switching, idle time is inserted using the LPTETP$'$ model.

   Otherwise, use API at the most beneficial job pair starting with the *initialsequence*. For each pair considered for switching, idle time is inserted using the LPTETP$'$ model.

9. If ($APIsolution < bestsolution$), then let $bestsolution = APIsolution$.

10. If ($switchcounter < 3$), go to Step 7.

11. If ($sequencecounter < 3$), go to Step 4.

12. Print, *bestsolution*.

The model LPTETP$'$ is the modified version of LPTETP model where the objective function is the summation of the unweighted earliness and tardiness penalties. Fry *et al.* (1990) stated that $HEUR_{FAR}$ found the optimal solution in 64 percent of the 192 random problems where the maximum number of jobs is 16. They did not give the computational complexity of $HEUR_{FAR}$, but it is obvious that $HEUR_{FAR}$ is an exponential algorithm due to the adjacent pairwise interchange argument at Step 8.

Chapter 3

Solution Procedures for $1|d_j|\Sigma(E_j + T_j)$

The vast majority of the total earliness and tardiness literature deals with the common due date case where all jobs have the same due date. This may be realistic for the production systems where jobs are processed as batches and every batch has an identical due date which is the common due date of the jobs in this batch. But when we consider production systems where jobs arrive independently and each has a predetermined due date, then models with distinct due dates become more realistic and applicable. Unfortunately, total earliness and tardiness problem with distinct due dates, even on a single machine, is shown to be $\mathcal{NP}$-hard. Hence, it is most unlikely that a polynomial algorithm will ever be developed for this problem. Due to its difficult nature, there is not much work about this problem in the literature. There exists two heuristics and two branch-and-bound algorithms in the literature with a few dominance properties and some lower bounding schemes.

The difficulty of the total earliness and tardiness problems with distinct due dates arises due to the possibility of idle time insertion between jobs. If it is known that the idle time insertion cannot improve objective function value, then we can limit our search only to the *permutation schedules*. This, in the worst case, results with $n!$ feasible solutions. But if inserted idle time becomes beneficial for a scheduling problem, then we have a larger feasible solution set. Furthermore, the non-regular performance measure of the problem $1|d_j|\sum(E_j + T_j)$ prevents the application of many important theorems on job ordering, particularly Emmons' (1969) and Lawler's (1977), in order to make optimization techniques such as branch-and-bound and dynamic programming more efficient. Hence, besides an optimizing algorithm, we also look for heuristics to obtain good solutions to this problem.

One of the purposes of this study is to develop both an optimizing algorithm and a heuristic algorithm for the problem $1|d_j|\sum(E_j + T_j)$ and test the results on a large set of random problems. The rest of this chapter is organized as follows[‡]. In the next section, we introduce a *dynamic programming* formulation for the problem $1|d_j|\sum(E_j+T_j)$ as an optimizing algorithm. This formulation is the first dynamic programming formulation developed for the problem $1|d_j|\sum(E_j + T_j)$ which allows idle time insertion in the schedule to the best of our knowledge. Unfortunately, this formulation leads to an exponential algorithm in the worst case. Considering this fact, we developed an *incomplete dynamic programming* algorithm by approximating the original dynamic programming formulation in Section 3.2. This incomplete dynamic programming algorithm forms the core of the heuristic we developed for the problem. In Section 3.3, the details of the heuristic algorithm are given including some basic dispatching rules and dominance properties. Section 3.4 reports on computational experience with the heuristic algorithm which is followed by some concluding remarks in Section 3.5.

[‡]This chapter draws heavily on Oğuz, Morin and Dinçer (1992)

3.1 A Dynamic Programming Algorithm

Since problem $1|d_j|\sum(E_j + T_j)$ is $\mathcal{NP}$-hard, it is most unlikely that any polynomial algorithm will ever be developed for this problem. Considering this result, we develop a dynamic programming algorithm for problem $1|d_j|\sum(E_j + T_j)$. This dynamic programming algorithm later forms the basis for the heuristic developed for the problem $1|d_j|\sum(E_j + T_j)$.

Dynamic programming has long been known to be a particular approach to optimization for scheduling problems. Dynamic programming is a away of looking at a problem which may contain a large number of interrelated decision variables so that the problem is regarded as if it consisted of a sequence of problems, each of which requires the determination of only one (or a few) variables. The basis of transforming a large problem with n variables to a set of smaller problems with a much smaller number of variables than n is known as the *principle of optimality.* It was originally proposed by Bellman (1957) and a simpler rendition of this principle can be stated as: Every optimal policy consists only of optimal sub-policies. An important advantage of dynamic programming is not being affected by the integrality constraints imposed on the variables of a problem. On the contrary, the requirement of having all of the variables as integers greatly simplifies the computational process in dynamic programming. There are, however, certain limitations to the use of dynamic programming. The principal one is the 'curse of dimensionality', i.e., the extremely fast growth of storage (and time) requirements, as the problem size increases. The numerical computation and storage requirements increase further in problems with two or more state variables rather than a single-state variable or in problems in which the number of states grows exponentially as the sequential decision process progresses.

Dynamic programming is applied to different sequencing problems by Lawler and Moore (1969). The underlying assumption for the single machine sequencing problems Lawler and Moore (1969) considered is that a set of n jobs are to be processed without any idle time in the schedule. Furthermore, their dynamic programming formulation carries out a search over all subsets of the states, that is, the time complexity of this dynamic programming is in $\mathcal{O}\ (2^n T)$ where T is described as a sufficiently large number. Later, Cheng (1990b) has presented a dynamic programming approach for the problem $1|d_j|\sum(E_j+T_j)$ with the restriction of no idle time insertion in the schedule. The computational requirements of this dynamic programming solution method is in the order of $n\,2^{n-1}$, hence it is an exponential algorithm. Later, Ibaraki and Nakamura (1990) have developed an alternative dynamic programming formulation for the same problem with the same restriction. The main advantage of their formulation is to use successive sublimation dynamic programming for state-space relaxations. The basics of successive sublimation dynamic programming is to execute a series of dynamic programming recursions, such that the underlying state-space is progressively refined at each iteration, until an exact optimal sequence of jobs is computed. Their computational experiences have shown that problem instances of up to $n = 35$ can be successfully solved with this method.

In this study again the development of an dynamic programming formulation is considered for the problem $1|d_j|\sum(E_j+T_j)$. The main contribution of this work is the elimination of the underlying assumption of the previous research. That is, we do not prohibit the insertion of idle time in the schedule. Relaxation of this assumption makes the problem harder because the completion times of the jobs can no longer be determined easily as the sum of the processing times of the jobs scheduled so far. In order to capture the true nature of the problem, we have to include a new variable denoting the idle time between any two jobs in the dynamic programming formulation. In spite of increasing the number of decision variables in the formulation, we expand the state-space in order to handle the insertion of idle times in the following dynamic programming formulation we developed. We ponder a job assigned to a position as a state rather than simply considering jobs. The main logic behind the dynamic programming formulation we developed is to look at the value of the performance measure by assigning every job to each position separately at every time point. Hence, the formulation takes into account all possible outcomes of the solution space.

In the functional equations of the dynamic programming, we will use the following notation in addition to those given in the previous chapters:

$f(i,j,t)$: the minimum attainable value of the penalty function for the partial schedule of $\sigma'_j \cup \{J_j\}$ when job $J_j \in \mathcal{J} \setminus \mathcal{J}_{\sigma'_j}$ is scheduled at position i and C_j being less than or equal to time t.

$\mathcal{J}$: the set of jobs that form a complete schedule.

$\mathcal{J}_{\sigma'_j}$: the set of jobs scheduled before job J_j that form a partial schedule for positions from 1 to $i-1$. These jobs will be referred as the leading jobs of job J_j.

i : position index, $i = 1, 2, \ldots, n$.

j : job index, $j = 1, 2, \ldots, n$.

t : time index, $t = 1, 2, \ldots, t_{max}$.

Then following functional equations of dynamic programming solve problem $1|d_j|\sum(E_j+T_j)$:

$$f(i,j,t) = \min\{\min_{J_j \neq J_k \in \mathcal{J},\ J_j \neq J_l \in \mathcal{J}_{\sigma'_k}} \{f(i-1,k,t-p_j)\} + |d_j - t|, f(i,j,t-1)\}$$

$$\text{for } i > 0, \quad j > 0, \quad t \geq 0$$

with the boundary condition

$$f(0,j,t) = 0 \quad \forall\, j, t \geq 0$$

and

$$f(i,j,t) = \infty \quad \text{if } \exists\, i < 0, \text{ or } \exists\, j < 0, \text{ or } \exists\, t \leq 0.$$

Furthermore, the optimal value of the objective function for problem $1|d_j|\sum(E_j+T_j)$ is

$$z^* = \min_{J_j \in \mathcal{J}}\{f(n,j,t_{max})\}.$$

In this formulation, job J_j at position i constitutes a *state*, and each time point t is a *stage*. Hence, we have a problem with n^2 *states* and t_{max} *stages* where

$$t_{max} = \max_{J_j \in \mathcal{J}}\{d_j\} + \sum_{j=1}^{n} p_j - \min_{J_j \in \mathcal{J}}\{p_j\}.$$

t_{max} gives the maximum possible time to be considered for any problem instance. This says that if the job with the maximum due date completes at its due date, then the time needed for this job should be equal to its due date and this is included as the maximum of the due dates. Then, all the other jobs will be tardy, and we need to include the summation of the processing times of the remaining jobs. This is achieved by adding the total processing times minus the minimum processing time to the maximum due date. We subtract the minimum processing time because of the on-time job. It is obvious that this schedule gives the longest time period to be considered for the problem and it is not meaningful to extend t_{max} beyond what is specified.

When scheduling job J_j, the minimization of the recursive function over jobs other than job J_j is required in order to prevent an infeasible schedule. Indeed, this is not enough to prevent an infeasible schedule because job J_k in position $i-1$ does not store the information about the jobs scheduled before job J_k. Hence, during this minimization, after finding job J_k that minimizes the recursive function, it must be checked whether all of the jobs scheduled before job J_k are different than job J_j. In fact, while searching for job J_k, the following must hold in order to schedule job J_j in position i:

$$J_j \neq J_l, \qquad \forall\, J_l \in \mathcal{J}_{\sigma'_k}.$$

Suppose the job J_j appears in the partial schedule of σ'_k. Then, another partial schedule will be formed for job J_k by replacing job J_j in σ'_k with job J_p where $p \in \sigma - \sigma'_k$ for which job J_p gives the minimum value of the functional equation for job J_k at position $i-1$ among all jobs in $\mathcal{J} \setminus \mathcal{J}_{\sigma'_k}$. Now, the above condition holds. After calculating the value of the recursive function for job J_k with this new partial schedule, it should be compared with the next minimum value of the functional equation and the minimum one selected.

From the above discussion, it is obvious that the partial schedules must be stored for each job scheduled to a position and this leads to excessive space requirements as n increases. Furthermore, the time complexity of the algorithm grows exponentially as n increases due to the step at which the feasibility of scheduling a job is checked and if any infeasibility exists, then attaining the feasibility again.

3.2 An Incomplete Dynamic Programming Algorithm

One of the most natural approaches to reduce problem dimension is to use some form of (usually lower-dimension) approximation. Typically, the 'larger' original dynamic program is approximated by a 'smaller' dynamic program, in which the state and/or decision spaces are subsets of those of the original dynamic program. One approach from approximation theory which has been used to reduce problem dimension in dynamic programming is to deal with a smaller set of states by discarding some of the states as mentioned in Morin (1979).

In our problem, we also look for a way to avoid the curse of dimensionality of the original dynamic programming formulation given in Section 3.1. The simplest way of doing so for our problem is to delete some of the states from our consideration and restrict the problem to a smaller set of states. Instead of the check and replacement mentioned in Section 3.1, we just form the leading jobs of each job at each stage. Then if we encounter with the newly scheduled job among these leading jobs at the checking step, we do not consider that job for that position by setting the value of the functional equation at that stage to infinity. This means that the newly scheduled job will never be considered for scheduling at that position and at that time. Since this will never lead to a schedule with a job assigned to more than one position, it is obvious that we can obtain a feasible schedule at the end. It is obvious that this approximation truncates the solution space and decreases the order of the running time of the algorithm since it eliminates the consideration of all subsets of the job set.

Such an approximation of a dynamic programming algorithm is called *incomplete dynamic programming* (IDP) in the literature (Erlenkotter 1975 and Morin 1979), since it consists of an approximation to the checking and replacement procedure. The time complexity function of this incomplete dynamic programming is $\mathcal{O}\ (n^4 T)$ since i and j are both in $\mathcal{O}\ (n)$, t is in $\mathcal{O}\ (T)$ and the inside minimization step over all jobs is in $\mathcal{O}\ (n^2)$. Hence, this is a pseudo-polynomial time algorithm (Garey and Johnson 1979).

For problems with small n, this approximation gives results very near to optimum. But as n increases, the deviation from the optimum solution by this approximation increases, as well. To overcome this difficulty, we incorporated some other procedures into the heuristic which are explained in the next section in detail. In order to exemplify above discussions, consider the following scheduling problem:

Example: The problem instance is given as follows for $n = 7$.

j	1	2	3	4	5	6	7
p_j	40	63	68	79	88	90	151
d_j	160	252	272	316	352	360	604

Let σ denotes the sequence of 7-jobs whereas $\mathcal{C}_\sigma$ and $\mathcal{P}_\sigma$ be the set of the completion times and the set of the penalties of jobs (negative for earliness and positive for tardiness) corresponding to the sequence σ, respectively.

The optimal solution of the above problem instance has an objective function value of $z^* = 380$ where $\mathcal{C}_\sigma = \{120, 160, 223, 302, 390, 480, 631\}$ and $\mathcal{P}_\sigma = \{-152, 0, -29, -14, 38, 120, 27\}$ for $\sigma = (3, 1, 2, 4, 5, 6, 7)$.

In order to obtain this optimal solution, IDP should find the leading job of job

J_1 at position $i = 2$ at time $t = 160$ as the job J_3. But when the corresponding functional equation is evaluated, we have:

$$\begin{aligned}
f(2,1,160) &= \min\{\min_{k\in\{2,3,4,5,6,7\}}\{f(1,k,120)\} + |160-160|, f(2,1,159)\} \\
&= \min\{132+0, \\
&\quad \min\{\min_{k\in\{2,3,4,5,6,7\}}\{f(1,k,119)\} + |159-160|, f(2,1,158)\}\} \\
&= \min\{132, \min\{133+1, 136\}\} \\
&= \min\{132, 134\} = 132
\end{aligned}$$

with $k = 2$.

If the functional equation was evaluated at the previous position at time $t = 120$ for job J_2 and job J_3, we see that $f(1,2,120) = 132$ and $f(1,3,120) = 152$. Hence, IDP will select job J_2 as the leading job of job J_1 at position $i = 2$ at time $t = 160$. At the next state, while computing the value of $f(3,2,223)$, IDP will find out that $k = 1$ minimizes the functional equation. That is, the leading job of job J_2 at position $i = 3$ at time $t = 223$ is job J_1. At the checking step of the leading jobs of job J_1, IDP will encounter with job J_2 and will set $f(3,2,223) = \infty$. So IDP will construct the feasible schedules on the partial schedule of $\sigma'_j = (2,1)$ for $j = 3,4,5,6,7$ at positions later than the position $i = 2$. Hence, it will never find the optimal solution for this problem instance.

3.3 A Heuristic Algorithm

In the heuristic that we developed for problem $1|d_j|\sum(E_j + T_j)$ $(HEUR)$, we start with the incomplete dynamic programming algorithm mentioned in previous section to obtain an initial solution. Further, we try three simple rules for the problem to obtain other initial sequences. These are EDD, EST and SPT rules. After obtaining the optimal schedules for these three initial sequences by utilizing the polynomial time algorithm of Garey *et al.* (1988) $OPTSCH|\sigma_{GTW}$, we apply an interchange procedure to all four schedules. In this procedure, first we select the job with the highest tardiness penalty and swap it with the job which is adjacent to it from left, if it is beneficial. Then, in the same way, we select the job with the highest earliness penalty and swap it with its adjacent job from the right, if it is beneficial. If a new sequence is obtained, that is if any two jobs are interchanged, we have to find the optimal schedule for this new sequence. Consequently, we select the schedule with the minimum total penalty among these four schedules. At the next step, we apply five dominance properties to the selected schedule. These dominance properties are well known for the problem $1|d_j|\sum(E_j+T_j)$ (Kim and Yano 1987, Fry *et al.* 1990, Hoogeveen 1992) and they are applied to each pair of adjacent jobs in the schedule. The formal description of four dominance properties are given below. The fifth dominance property is the same with Proposition 2.8 in Section 2.6.4.2.

Proposition 3.1 *Suppose job J_{j_1} and job J_{j_2} are adjacent pair of jobs in a sequence. If*

$$\max\{C_{j_1}, C_{j_2}\} \leq \min\{d_{j_1}, d_{j_2}\},$$

that is if both jobs are early jobs, then a sequence in which,

- *if $p_{j_1} > p_{j_2}$, job J_{j_1} should precede job J_{j_2},*
- *if $p_{j_1} < p_{j_2}$, job J_{j_2} should precede job J_{j_1}.*

is better.

Proof: Let $ET_{j_1j_2}$, $ET_{j_2j_1}$ and c_{max} be as defined in Section 2.6.4.2. Then,

$$ET_{j_1j_2} = d_{j_2} - d_{j_1} + 2\,(d_{j_1} - c_{max}) + p_{j_2},$$

$$ET_{j_2j_1} = d_{j_2} - d_{j_1} + p_{j_1} + 2\,(d_{j_1} - c_{max}),$$

$$ET_{j_1j_2} - ET_{j_2j_1} = p_{j_2} - p_{j_1}.$$

Hence the result follows. ■

Proposition 3.2 *Suppose job J_{j_1} and job J_{j_2} are adjacent pair of jobs in a sequence. If*

$$\min\{S_{j_1}, S_{j_2}\} \geq \max\{d_{j_1}, d_{j_2}\},$$

that is if both jobs are tardy jobs, then a sequence in which,

- *if $p_{j_1} < p_{j_2}$, job J_{j_1} should precede job J_{j_2},*
- *if $p_{j_1} > p_{j_2}$, job J_{j_2} should precede job J_{j_1}.*

is better.

Proof:

$$ET_{j_1j_2} = 2\,(c_{max} - d_{j_2}) + 2\,p_{j_1} + (d_{j_2} - d_{j_1}) + p_{j_2},$$
$$ET_{j_2j_1} = 2\,(c_{max} - d_{j_2}) + 2\,p_{j_2} + (d_{j_2} - d_{j_1}) + p_{j_1},$$
$$ET_{j_1j_2} - ET_{j_2j_1} = p_{j_1} - p_{j_2}.$$

Hence the result follows. ■

Proposition 3.3 *Suppose job J_{j_1} and job J_{j_2} are adjacent pair of jobs in a sequence. If*

$$\min\{S_{j_1}, S_{j_2}\} + p_{j_1} > d_{j_1}$$

and

$$\min\{S_{j_1}, S_{j_2}\} + p_{j_1} + p_{j_2} < d_{j_2}$$

then a sequence in which job J_{j_1} should precede job J_{j_2}, regardless of the size of their processing times, is better.

Proof: If the conditions given in the proposition hold, then job J_{j_1} is tardy and job J_{j_2} is early in that sequence. If an interchange of these two jobs takes place, it is obvious that the tardiness of job J_{j_1} and the earliness of job J_{j_2} will increase which will not improve the objective function value. ■

Proposition 3.4 *Suppose job* J_{j_1} *and job* J_{j_2} *are adjacent pair of jobs in a sequence. If* $p_{j_1} = p_{j_2}$ *and* $d_{j_1} \le d_{j_2}$*, then job* J_{j_1} *should precede job* J_{j_2}*.*

Proof: Trivial. ■

The following is a brief statement of our heuristic, *HEUR*.

1. Find the first initial schedule by applying IDP algorithm.
2. Obtain three additional initial sequences by using EDD, EST and SPT rules and determine their optimal schedules by applying $OPTSCH|\sigma_{GTW}$.
3. Apply interchange procedure to these four schedules obtained from step 1 and step 2. If any of the sequences changes, find the optimal schedule for the new sequence by applying $OPTSCH|\sigma_{GTW}$.
4. Choose the schedule with the minimum total penalty among the schedules obtained in step 3.
5. Apply the five dominance properties to the selected schedule and stop.

3.4 Computational Experience

For our experiment, 1375 problems have been generated randomly with the method used in Fisher (1976). In this method, for each job J_j an integer processing time p_j was generated from the uniform distribution $[1,100]$. Due date generation is based on the sum of the processing times of all jobs ($P = \sum_{j=1}^{n} p_j$) and two parameters that determine the problem 'hardness'. These two parameters are the *relative range of due dates* (R) and the *average tardiness factor* (T). The average tardiness factor, T, is a coarse measure of the proportion of jobs that might be expected to be tardy in an arbitrary sequence. The relative range of due dates, R, determines the priority of jobs. When the relative range of due dates is tight, the priority of all jobs would tend to rise at about the same time, making it difficult to discriminate between urgent and not-so-urgent jobs. So, R controls the range of the due date distribution. Having computed P and chosen R and T, an integer due date d_j was generated from the uniform distribution $[P(1-T-R/2), P(1-T+R/2)]$ for each job J_j. In our computational experiment, we choose R and T to start from 0.2 and increase to 1.0 with 0.2 increments. Furthermore, we truncate the due date d_j to zero if it comes out to be a negative value. We select eleven values to determine the problem size where $n = 8, 10, 12, 14, 16, 18, 20, 25, 30, 35, 40$.

As a result each problem set is defined with three parameters, namely R, T, and n. For each value of n, five problems were generated for each of the 25 pairs of values of R and T, yielding 125 problems for each value of n and a total of 1375 problems.

The algorithm was coded in C and run on Sun Microsystems' Sun-4(SPARC) workstations. The optimal solutions for the test problems obtained from the package program CPLEX Mixed Integer Optimizer which runs on the same computer systems. But CPLEX can not find optimum, even incumbent solutions for the problems where n is greater than 18. Hence, the results are given in two parts. The first part includes the problems with $n = 8, 10, 12, 14, 16, 18$ where the performance of $HEUR$ is compared with the optimal solution. The performance of $HEUR$ for the problems with $n = 20, 25, 30, 35, 40$ is compared with that of Kim and Yano (1987) ($HEUR_{KY}$) and Fry *et al.* (1990) ($HEUR_{FAR}$) in the second part. It should be noted here that due to the discussion in Section 2.6.4.3, we used a modified version of $HEUR_{FAR}$ described in that section in the sense that the incorrect dominance property that Fry *et al.* (1990) considered is eliminated in the heuristic used in our experiment.

Part I:

The results for $n = 8, 10, 12, 14, 16, 18$ are summarized in the following tables. In Table 3.1, we present the average and maximum relative errors of $HEUR$ together with the percentage of those 125 random problems that $HEUR$ finds out the optimal solution for every value of n in the last column. The other columns show the performance of IDP, EDD, EST and SPT for the same criteria. The relative error for any rule is given as $(z_{RULE} - z^*)/z^*$ and the average and maximum relative errors are found over 125 random problems for every n value.

Table 3.1: Average (AVG) and maximum (MAX) relative errors together with percentage of optimal solutions found by different rules for different n values.

n		IDP	EDD	EST	SPT	*HEUR*
8	AVG	0.151	0.289	0.577	1.421	0.043
	MAX	1.283	0.963	1.810	9.914	0.620
	% OPT	34.4	6.4	1.6	2.4	57.6
10	AVG	0.278	0.336	0.590	1.581	0.067
	MAX	3.378	1.329	1.884	11.162	0.383
	% OPT	16.8	1.6	0.8	0.8	35.2
12	AVG	0.253	0.387	0.615	1.769	0.083
	MAX	1.550	1.165	1.483	17.225	0.619
	% OPT	13.6	0	0	0.8	28.8
14	AVG	0.372	0.424	0.589	1.916	0.100
	MAX	6.525	1.263	1.533	12.789	0.689
	% OPT	10.4	0.8	0	0	16.8
16	AVG	0.375	0.441	0.617	2.110	0.104
	MAX	6.144	1.129	1.444	16.456	0.363
	% OPT	3.2	0	0	0	10.4
18	AVG	0.360	0.445	0.609	1.504	0.118
	MAX	1.213	0.958	1.256	9.546	0.444
	% OPT	4	0	0	0	6.4
AVG(over 750 problems)		0.298	0.387	0.560	1.717	0.086
% OPT(over 750 problems)		0.137	0.015	0.004	0.007	0.259

The solution of *HEUR* is on average 8.6 % over the optimal solution which is much better than any of the other rules. In addition, *HEUR* finds the optimal solution in more than 25 % of the time. As a result, our heuristic outperforms any of the other rules both in terms of the average and the maximum relative errors as well as the percentage of optimal solutions found. Also, IDP itself performs better than other three rules on the average. It is interesting to note that IDP has lower average relative error compared to that of EDD, EST, and SPT rules, although it has higher maximum relative error. This indicates that IDP is much better than any of the other rules in most of the problems. This conclusion is manifested by tables 3.2, 3.3 and 3.4. The frequencies given in these tables indicate the distribution of the relative errors for different rules. In summary, these tables show that *HEUR* and IDP finds optimal solution in more cases than other rules. Another point in the computational study to stress is that the relative errors increase for any of the rules as well as for our heuristic as the number of jobs increases.

Table 3.2: Frequency distribution of solution of different rules for $n = 8$ and $n = 10$.

	$n = 8$					$n = 10$				
interval	IDP	EDD	EST	SPT	*HEUR*	IDP	EDD	EST	SPT	*HEUR*
0	43	8	2	3	72	21	2	1	1	44
0.1	27	20	4	8	31	37	15	5	13	51
0.2	23	23	7	7	13	13	20	10	8	15
0.3	7	23[a]	14	11	8	12	30	14	5	11
0.4	10	19	19	6	0	9	20	12	8	4
0.5	8	9	20	5	0	16	13	15	3	0
0.6	2	8	10	9	0	4	10	18[b]	5	0
0.7	2	8	11	2	1	4	5	16	6	0
0.8	1	2	5	7	0	3	5	6	6	0
0.9	0	4	12	10	0	0	2	6	3	0
1	0	1	6	4	0	0	1	3	9	0
	2	0	15	53	0	6	2	19	58	0

[a]In 23 of 125 problems, the solution found by EDD rule lies between 21-30 % of the optimal solution.

[b]In 18 of 125 problems, the solution found by EST rule lies between 51-60 % of the optimal solution.

Table 3.3: Frequency distribution of solution of different rules for $n = 12$ and $n = 14$.

	$n = 12$					$n = 14$				
interval	IDP	EDD	EST	SPT	*HEUR*	IDP	EDD	EST	SPT	*HEUR*
0	17	0	0	1	36	13	1	0	0	21
0.1	42	9	1	9	51	35	8	1	10	59
0.2	15	17	1	8	23	17	9	6	6	22
0.3	10	23	15	6	9	7	19	13	9	15
0.4	9	22	18	11	4	12	24	10	4	6
0.5	6	21	23	9	0	10	23	25	4	0
0.6	9	14	14	3	1	6	14	18	3	1
0.7	10	11	10	4	1	6	12	14	10	1
0.8	3	1	8	4	0	5	10	9	4	0
0.9	0	2	11	2	0	4	3	11	5	0
1	0	1	10	5	0	1	0	9	5	0
	4	4	14	63	0	9	2	9	65	0

Table 3.4: Frequency distribution of solution of different rules for $n = 16$ and $n = 18$.

	$n = 16$					$n = 18$				
interval	IDP	EDD	EST	SPT	*HEUR*	IDP	EDD	EST	SPT	*HEUR*
0	4	0	0	0	13	5	0	0	0	8
0.1	38	7	1	9	63	28	4	0	10	37
0.2	15	8	2	6	25	7	9	3	4	28
0.3	17	20	13	8	16	8	8	5	10	11
0.4	12	27	20	7	8	6	14	11	2	6
0.5	12	15	17	2	0	7	19	7	5	1
0.6	6	20	14	4	0	8	17	20	2	0
0.7	6	12	17	5	0	4	12	13	4	0
0.8	2	9	13	7	0	6	6	12	0	0
0.9	2	1	7	8	0	2	1	11	5	0
1	3	5	5	3	0	4	1	5	4	0
	8	1	16	66	0	6	0	4	45	0

In Table 3.5, we give the average relative errors of *HEUR* for five problems generated for each combination of R and T values for different values of n. From Table 3.5, it is seen that the easiest problems belong to $T = 1.0$ for every value of R. As T decreases the relative error of the heuristic increases. But the worst

Table 3.5: Average relative errors of 5 problems for every pair of R and T values for every n value for *HEUR*.

		n					
R	T	8	10	12	14	16	18
0.2	0.2	0.096	0.254	0.183	0.152	0.140	0.202
0.2	0.4	0.134	0.187	0.108	0.184	0.147	0.219
0.2	0.6	0.077	0.118	0.088	0.237	0.205	0.176
0.2	0.8	0.023	0.027	0.015	0.020	0.028	0.037
0.2	1.0	0.000	0.000	0.001	0.000	0.002	0.001
0.4	0.2	0.070	0.110	0.141	0.088	0.150	0.163
0.4	0.4	0.182	0.170	0.181	0.252	0.263	0.182
0.4	0.6	0.012	0.036	0.052	0.105	0.097	0.112
0.4	0.8	0.015	0.027	0.042	0.011	0.031	0.040
0.4	1.0	0.000	0.002	0.000	0.009	0.006	0.004
0.6	0.2	0.104	0.039	0.174	0.100	0.123	0.179
0.6	0.4	0.027	0.113	0.149	0.175	0.112	0.208
0.6	0.6	0.075	0.028	0.078	0.105	0.160	0.191
0.6	0.8	0.012	0.063	0.031	0.125	0.071	0.048
0.6	1.0	0.007	0.010	0.003	0.033	0.037	0.004
0.8	0.2	0.044	0.047	0.118	0.179	0.193	0.185
0.8	0.4	0.031	0.031	0.126	0.076	0.064	0.560
0.8	0.6	0.021	0.050	0.091	0.116	0.103	0.109
0.8	0.8	0.041	0.024	0.101	0.065	0.055	0.047
0.8	1.0	0.008	0.058	0.066	0.013	0.024	0.023
1.0	0.2	0.072	0.000	0.107	0.087	0.073	0.081
1.0	0.4	0.000	0.091	0.002	0.158	0.123	0.154
1.0	0.6	0.003	0.044	0.082	0.093	0.125	0.106
1.0	0.8	0.010	0.097	0.014	0.084	0.185	0.163
1.0	1.0	0.007	0.058	0.131	0.035	0.079	0.056

performance is obtained when $T = 0.4$. It should be noted that as T decreases, the number of the early jobs increases. So, our algorithm works better in the situations where the majority of the jobs are tardy. This is because of the dominance properties used in the algorithm which are Emmons' like properties so they are powerful on tardy jobs. Furthermore, the computational experiences show that IDP fall short of scheduling early jobs correctly. On the contrary, IDP is successful in scheduling tardy jobs.

The solution time of *HEUR* is also compared with the solution time of CPLEX and the results are presented in Table 3.6. When we compare the average CPU time spent for solution, it is clear that CPLEX takes considerably longer time than *HEUR* when $n > 10$. The average CPU time of CPLEX is even greater than the maximum CPU time spent by *HEUR*. Solution time is an important factor in real-time applications when job numbers increase and in such a case our computational experience suggests that *HEUR* can be preferred instead of finding the optimal solutions.

Table 3.6: CPU times (in seconds) of the problems for different n values.

		n					
		8	10	12	14	16	18
CPLEX	Avg.CPU	0.775	6.107	46.654	904.849	2668.036	15949.593
HEUR	Avg.CPU	2.928	7.934	18.533	36.849	71.756	119.383
	Max.CPU	6.016	14.983	36.065	68.664	130.861	217.358

Part II:

Since we could not obtain the optimal solutions for $n = 20, 25, 30, 35, 40$ we compare the performance of $HEUR$ for these problems with the heuristics from the literature. Thus, after coding the heuristics of Kim and Yano (1987) and Fry *et al.* (1990), we compare the performance of $HEUR$ with these two heuristics. For this part of the computational experiment, we use the average ratio of the solution generated by the heuristic of Fry *et al.* (1990) to the solution generated by $HEUR$ ($HEUR_{FAR}/HEUR$) over 125 random problems for different n values, the percentage of those 125 random problems that $HEUR_{FAR}$ outperforms $HEUR$ (% $HEUR_{FAR}$), and similarly the average ratio of the solution generated by the heuristic of Kim and Yano (1987) to the solution generated by $HEUR$ ($HEUR_{KY}/HEUR$) over 125 random problems for different n values and the percentage of those 125 random problems that $HEUR_{KY}$ outperforms $HEUR$ (% $HEUR_{KY}$). Table 3.7 reports our computational experience. The average

Table 3.7: Computational Experience of Part II.

n		$HEUR_{KY}$	$HEUR_{FAR}$
20	AVG. ratio to $HEUR$	2.241	0.903
	%	0	88.8
25	AVG. ratio to $HEUR$	2.524	0.885
	%	0	96
30	AVG. ratio to $HEUR$	2.511	0.876
	%	0	92.8
35	AVG. ratio to $HEUR$	2.499	0.860
	%	0.8	95.2
40	AVG. ratio to $HEUR$	2.522	0.823
	%	3.2	100
Avg. of AVG(over 625 problems)		2.459	0.869
Avg. of % (over 625 problems)		0.8	94.56

ratio of $HEUR_{KY}/HEUR$ is 2.459 and the average value of % $HEUR_{KY}$ is 0.8 which suggests that $HEUR$ outperforms the heuristic of Kim and Yano (1987).

But the average ratio of $HEUR_{FAR}/HEUR$ is 0.870 and the average value of % $HEUR_{FAR}$ is 94.56, which suggests that $HEUR$ is not as good as the heuristic of Fry *et al.* (1990). However it should be noted that the heuristic of Fry *et al.* (1990) is an exponential time algorithm due to the adjacent pairwise interchange procedure which forms the basis of this algorithm. Hence for the problems with large n, it may be advantageous to utilize $HEUR$ if the solution time is one of the concerns.

3.5 Concluding Remarks

In this chapter, first a dynamic programming formulation to minimize total unweighted earliness and tardiness penalties on a single machine is developed. This formulation leads to an incomplete dynamic programming and a heuristic based on the incomplete dynamic programming is presented in the following sections. The heuristic also involves a simple interchange procedure and five well known dominance properties.

The heuristic was tested in two parts. In the first part, there are 750 randomly generated problems for problem sizes $n = 8, 10, 12, 14, 16, 18$. For this set of problems, the optimal solutions are obtained from the solution of the mixed-integer program for the problem $1|d_j|\sum(E_j + T_j)$. The average relative error of our heuristic is 8.6% and it outperforms all other dispatching rules used in the study. Furthermore, the percentage of time the optimal solutions found by the heuristic is over 25%. For these problems, two problem parameters, namely relative range of due dates and average tardiness factor, are also changed from 0.2 to 1.0 with 0.2 increments. It is seen that our heuristic performs well when the value of average tardiness factor is large for any value of the relative range of due dates.

In the second part of the study, we generated 625 random problems for $n = 20, 25, 30, 35, 40$ for which we could not obtain the optimal solutions. These set of problems are evaluated with respect to two well known heuristics from the literature. The computational experience shows that our heuristic outperforms the heuristic of Kim and Yano (1987) in all respects. However, the performance of our heuristic is not better than that of Fry *et al.* (1990) considering the relative errors. But since the latter one is an exponential time algorithm, this is what we expect from the computational experiments.

The development of an exact *dynamic programming* which is still pseudo-polynomial for this class of scheduling problems remains as a further research topic. Since this class of problems are known to be $\mathcal{NP}$-hard in the ordinary sense, the development of such a dynamic programming may lead to fully-polynomial approximation schemes.

Chapter 4

$1|d_j|\Sigma(E_j + T_j)$ with Special Structures for Distinct Due Dates

Since the problem $1|d_j|\sum(E_j + T_j)$ is $\mathcal{NP}$-hard in its general form, it may be beneficial to deal with some special cases which will lead to efficient solution procedures for the general case. Considering this fact, we deal with the problems in which we can restrict the structure of the due dates by applying special due date assignment rules. These different due date structures can be viewed as the priority rules for the jobs since they determine the jobs' urgency in the shop.

Indeed, these rules are studied in the literature but as the due date assignment rules in the multi-machine environments (see for example Baker (1984)). In recent years, some research have been evolved that analyze different due date assignment rules for the single machine scheduling problems in which one of the aims is to determine the optimal due dates, such as Cheng (1984) and Quaddus (1987). Since the optimality criteria like earliness and tardiness depend on due dates, the assignment rule of the due dates affects the efficiency of the solution procedures developed. Conway *et al.* (1967) reported four rules to assign due dates in a job shop environment where as Baker (1984) extended these rules to more complex ones.

The importance of the due date assignment rules for the problem that we deal with is that, as Baker and Scudder (1990) states, these type of models may be structurally less complicated than the general model with distinct due dates. Furthermore, the solution procedures provided for these restricted models may be helpful in the analysis of the general model. Baker and Scudder (1990) have given the following two rules for determining the distinct due dates which relate to the processing times:

- Equal-Slack(SLK): $d_j = p_j + q$
- Total-Work-Content(TWC): $d_j = k\,p_j$

As its name implies, the SLK rule gives an equal slack for every job, hence their risk to be tardy and early is equal whereas their due dates differ according to their processing times. As it will be seen in Section 4.1, this rule provides a nice structure of the problem in certain cases. The second rule, TWC rule, gives to a job an allowance which is a multiple of its processing time. Hence, a job can wait $(k-1)\,p_j$ time units before it is processed and to be completed on-time. These rules result in reasonable and attainable due dates since they provide a greater allowance for a job with larger processing time among all the jobs.

In the rest of this chapter, we consider these two rules and the results are presented in the following sections. In Section 4.1[‡] , we study the structure of the problem $1|d_j|\sum(E_j+T_j)$ when due dates are given according to SLK rule. The problem is analyzed in Section 4.1.1. We discuss that this problem can be classified as restricted and unrestricted according to a constraint on q. In Section 4.1.2, the equivalence of the unrestricted case to a polynomially solvable problem is given. We further analyze the restricted case of the problem

[‡]This section draws heavily on Oğuz and Dinçer (1992)

in section 4.1.3 and identify some optimality conditions. Finally, in Section 4.1.4 we state that the restricted case is $\mathcal{NP}$-hard.

In Section 4.2, we consider the second rule. Hence our problem is $1|d_j|\sum(E_j+T_j)$ where due dates are given according to the Total-Work-Content rule. In Section 4.2.1, we analyze the problem of $1|d_j|\sum(E_j+T_j)$ under TWC rule and we present several properties of this problem. In Section 4.2.2, we develop a polynomial time heuristic algorithm with a very good performance. The computational experience regarding this heuristic is given in Section 4.2.3.

4.1 Problem $1|d_j|\sum(E_j+T_j)$ with Equal-Slack Rule

The slack time of a job is its remaining allowance to finish its processing. Equal-Slack rule assigns an equal amount of slack time to every job J_j for processing. The intuitive justification for Equal-Slack rule is that all jobs have to start at a prespecified time in order to be completed at their due dates and they have the same urgency for being on-time. We will determine some properties of this special case in the following section where the problem is splitted to two cases as unrestricted and restricted. After showing the equivalence of the unrestricted case to a polynomially solvable problem, the $\mathcal{NP}$-hardness for the restricted case is presented.

4.1.1 Analysis of $1|d_j=p_j+q|\sum(E_j+T_j)$

We consider the problems in which distinct due dates are related to processing times according to Equal-Slack rule. This means that distinct due dates are given as $d_j = p_j + q,\ q > 0 \quad \forall\ j = 1,2,\ldots,n$. Hence our problem can be stated as $1|d_j = p_j + q|\sum(E_j+T_j)$ using our notation. Since this problem permits distinct due dates that relate to processing times, it allows a nice structure for the optimal solution in a special case where the problem is unrestricted.

For further analysis, the notation $\sum_{j=1}^{n}|C_j - d_j|$ will be used instead of $\sum_{j=1}^{n}(E_j + T_j)$ since minimizing the sum of unweighted earliness and tardiness penalties is equivalent to minimizing the sum of absolute deviation of completion times from respective distinct due dates. Furthermore, it is easy to observe the followings:

Proposition 4.1 *Minimizing* $|C_j - d_j|$ *is equivalent to minimizing* $|S_j - a_j|$.

Proof: From definition we know that $d_j = a_j + p_j$. Substituting this into $|C_j - d_j|$ yields the following:

$$|C_j - d_j| = |C_j - (a_j + p_j)| = |C_j - p_j - a_j|.$$

But from definition $S_j = C_j - p_j$. So by substituting this one obtains,

$$|C_j - d_j| = |S_j - a_j|.$$

Hence, minimizing $|C_j - d_j|$ is equivalent to minimizing $|S_j - a_j|$. ■

Proposition 4.2 *If* $d_j = p_j + q$ *then* $a_j = q \quad \forall\ j = 1, 2, \ldots, n.$

Proof: This result is obtained easily after substituting $d_j = p_j + q$ into $a_j = d_j - p_j$ as follows:

$$a_j = d_j - p_j = p_j + q - p_j = q.$$

■

Proposition 4.2 means that each job with $d_j = p_j + q$ have a common target starting time, namely q.

Theorem 4.1 $1|d_j = p_j + q|\sum(E_j + T_j)$ *is equivalent to* $1|a_j = q|\sum|S_j - a_j|$.

Proof: This follows directly from Proposition 4.1 and Proposition 4.2. ■

From above observations, it is seen that we are trying to minimize the sum of absolute deviation of starting times from a common target starting time. This problem is analogous to the one in which the sum of absolute deviation of completion times from a common due date is minimized. Hence, following the research for the second problem, we analyzed the common target starting time to determine the point where problem becomes unrestricted. We can say that a problem is unrestricted if the common target starting time is loose such that we can schedule enough jobs before q. That is, we have freedom to schedule some jobs before q. If common target starting time is tight, then we will not be able to schedule some of the jobs before q since the starting time of any schedule can not be less than zero. Thus, for a given set of jobs, if q is too tight, we encounter with the restricted version of the problem.

Formally, we define a problem unrestricted if $q \geq \vartheta$ where ϑ is defined as below assuming that the processing times are indexed in non-decreasing order:

$$\vartheta = \begin{cases} p_1 + p_3 + \cdots + p_{n-1} & \text{if } n \text{ is even} \\ p_2 + p_4 + \cdots + p_{n-1} & \text{if } n \text{ is odd} \end{cases}$$

Before going further, it is necessary to give the following notations which are similar to those given in Section 1.1 and modified for the common target starting time problem:

4.1.2 Analysis of $1|a_j = q \geq \vartheta|\sum|S_j - a_j|$

It is possible to show that an optimal solution to the problem $1|a_j = q \geq \vartheta|\sum|S_j - a_j|$ possesses the following properties:

Proposition 4.3 *There is no idle time between any two jobs in the optimal schedule.*

Proof: Assume to the contrary that there is an idle time in an optimal schedule of σ. This idle time can be either before q or after q. In the former case, it is obvious that the idle time occurs between two early jobs, say job J_{j_1} and J_{j_2}. Now, we can decrease the earliness of the jobs before job J_{j_1} and the job J_{j_1} itself by moving them till the starting time of the job J_{j_2}. Thus, we improve the value of the objective function. Contradiction. In the later case, similar to the first one, the idle time occurs between two tardy jobs, say again jobs J_{j_1} and J_{j_2}. Now, we can decrease the tardiness of the jobs after job J_{j_2} and job J_{j_2} itself by moving them till the completion time of the job J_{j_1}. Thus, we improve the value of the objective function. Again contradiction. ■

Proposition 4.4 *One job will start exactly on the target starting time q.*

Proof: Assume the contrary and consider a schedule of σ which has a job J_{j_1} which is in process at time q. That is in σ, there exists a job J_{j_1} such that $S_{j_1} < q$ and $C_{j_1} > q$ as shown in Figure 4.1. Let $p_{j_1}(b)$ be the amount of processing time

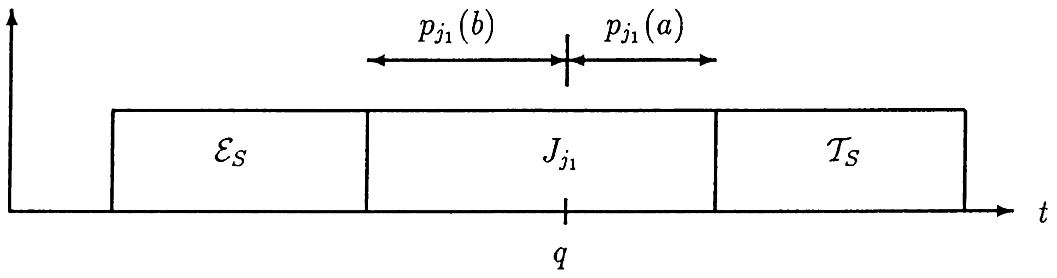


Figure 4.1: Schedule of σ for Proposition 4.4.

of job J_{j_1} before q and let $p_{j_1}(a)$ be the amount of processing time of job J_{j_1} after q. Clearly, either $|\mathcal{E}_S| < |\mathcal{T}_S|$ or $|\mathcal{E}_S| \geq |\mathcal{T}_S|$. First suppose that $|\mathcal{E}_S| < |\mathcal{T}_S|$. Then all jobs can be shifted to the left such that $C_{j_1} = q$. The change in $z(\sigma)$ due to this shift is

$$|\mathcal{E}_S|\,p_{j_1}(a) - |\mathcal{T}_S|\,p_{j_1}(a) + p_{j_1}(a) \leq 0.$$

If $|\mathcal{E}_S| \geq |\mathcal{T}_S|$, then all jobs can be shifted to the right such that $S_{j_1} = q$. The change in $z(\sigma)$ due to this shift is

$$|\mathcal{T}_S|\,p_{j_1}(b) - |\mathcal{E}_S|\,p_{j_1}(b) - p_{j_1}(b) < 0.$$

In either case the value of the objective function of the schedule of σ will be improved. Contradiction. ■

Proposition 4.5 *The optimal schedule is V-shaped around the job that starts at time q.*

Proof: Suppose the schedule of σ is not V-shaped but optimal. That is either $\mathcal{E}'_S$ is not in LPT order or $\mathcal{T}_S$ is not in SPT order. Assume the former one is true (the proof of the latter is similar). If $\mathcal{E}'_S$ is not in LPT order then there exists two adjacent jobs J_{j_1} and J_{j_2} such that $p_{j_1} < p_{j_2}$ and job J_{j_1} precedes job J_{j_2} as shown in Figure 4.2. Let σ' is obtained from σ by interchanging jobs J_{j_1} and J_{j_2}.

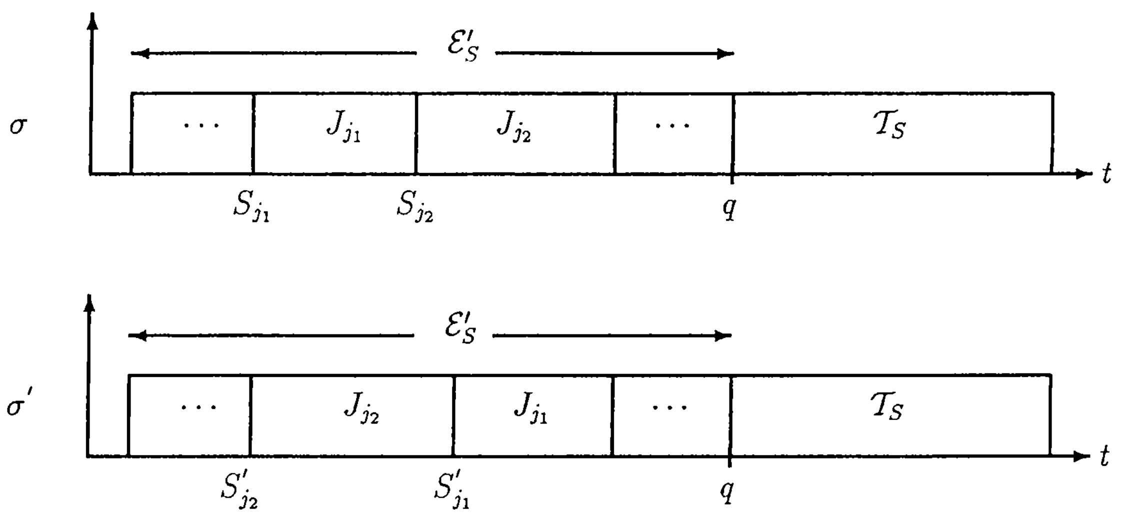


Figure 4.2: Schedule of σ for Proposition 4.5.

Clearly, such an interchange does not affect the penalty for any job other than jobs J_{j_1} and J_{j_2}. Then

$$z(\sigma') - z(\sigma) = (q - S'_{j_1}) - (q - S_{j_1}) + (q - S'_{j_2}) - (q - S_{j_2}).$$

Since $S'_{j_2} = S_{j_1}$, $S'_{j_1} = S_{j_1} + p_{j_2}$ and $S_{j_2} = S_{j_1} + p_{j_1}$, we have

$$z(\sigma') - z(\sigma) = p_{j_1} - p_{j_2} < 0.$$

Thus $z(\sigma') < z(\sigma)$, contrary to the assumption that σ is optimal. ■

With these optimality conditions, we can show that the problem $1|a_j = q \geq \vartheta|\sum|S_j - a_j|$ is equivalent to the problem $1|d_j = d \geq \delta|\sum|C_j - d_j|$.

It is clear that the objective function values of both problems are independent of the absolute values of q and d, since they are non-restrictive. Instead, the objective values are functions of the sequence of jobs and the order of the job which starts (or completes) at q (or d). Hence, let $f_S(\varepsilon,\pi) = \sum_{i=1}^{n}|S_{\pi(i)} - q|$ and $f_C(\tau,\sigma) = \sum_{i=1}^{n}|C_{\sigma(i)} - d|$ where ε and τ are such that $S_{\pi(\varepsilon)} = q$ and $C_{\sigma(\tau)} = d$.

Theorem 4.2 *Let σ and π be two sequences with $\sigma(i) = \pi(n-i+1)$. Arrange σ and π so that $C_{\sigma(\tau)} = d$ and $S_{\pi(\varepsilon)} = q$, where $n-\tau = \varepsilon$. Then $f_S(\varepsilon,\pi) = f_C(\tau,\sigma)$.*

Proof: Let σ and π be sequences of n jobs. If we let $\sigma(\tau)$ be the job with completion time equal to d, then

$$\begin{aligned} f_C(\tau,\sigma) &= p_{\sigma(2)} + 2\,p_{\sigma(3)} + \cdots + (\tau-1)\,p_{\sigma(\tau)} \\ &\quad + (n-\tau)\,p_{\sigma(\tau+1)} + (n-\tau-1)\,p_{\sigma(\tau+2)} + \cdots + p_{\sigma(n)}. \quad (4.1)\end{aligned}$$

Similarly, if we let $\pi(\varepsilon)$ be the job with starting time equal to q, then

$$\begin{aligned} f_S(\varepsilon,\pi) &= p_{\pi(1)} + 2\,p_{\pi(2)} + \cdots + \varepsilon\,p_{\pi(\varepsilon)} \\ &\quad + (n-\varepsilon-1)\,p_{\pi(\varepsilon+1)} + (n-\varepsilon-2)\,p_{\pi(\varepsilon+2)} + \cdots + p_{\pi(n-1)}. (4.2)\end{aligned}$$

When we let $n-\tau = \varepsilon$ and $\sigma(i) = \pi(n-i+1)$, the claim follows. ■

Theorem 4.2 shows that if σ is an optimal solution to $1|d_j = d \geq \delta|\sum|C_j - d_j|$, then π, as defined in the theorem, is an optimal solution to $1|a_j = q \geq \vartheta|\sum|S_j - a_j|$. Therefore, any algorithm which solves the problem $1|d_j = d \geq \delta|\sum|C_j - d_j|$, such as the algorithm proposed by Kanet (1981a) or by Bagchi *et al.* (1986), can be used to solve the problem $1|a_j = q \geq \vartheta|\sum|S_j - a_j|$.

Considering the equivalence of the problem $1|a_j = q \geq \vartheta|\sum|S_j - a_j|$ to the problem $1|d_j = d \geq \delta|\sum|C_j - d_j|$, we can state the following:

Proposition 4.6 *In an optimal schedule, the $(|\mathcal{E}'_S|+1)$th job in sequence starts at its target starting time, where $|\mathcal{E}'_S|$ is the smallest integer greater than or equal to $(n-1)/2$.*

Proof: Proof follows from the equivalence of the problem $1|a_j = q \geq \vartheta|\sum|S_j - a_j|$ to the problem $1|d_j = d \geq \delta|\sum|C_j - d_j|$ and Proposition 2.7 in Section 2.5.2. ■

4.1.3 Analysis of $1|a_j = q < \vartheta|\sum|S_j - a_j|$

When we consider the restricted version of the problem, we see that only the Proposition 4.3 holds, but others do not. Also, although a polynomial time algorithm exists for the unrestricted problem, this will not be the case for the restricted problem. Following the work of Hall *et al.* (1991), we describe several necessary optimality conditions for the problem $1|a_j = q < \vartheta|\sum|S_j - a_j|$.

We define a unique job J_α in a sequence such that it starts processing before q and completes at or after q. Hence, only for job J_α, $S_\alpha + p_\alpha = q$ if a job completes at q, and $S_\alpha + p_\alpha > q$ if no job does so. It is obvious that if $q \leq \min_j\{p_j\}$, then an SPT sequence of the jobs specifies a solution.

For the problem $1|a_j = q < \vartheta|\sum|S_j - a_j|$ Proposition 4.4 may not hold as shown by the instance $n = 3$, $p_1 = 5, p_2 = 6, p_3 = 10$ where $q = 4$ with a unique optimal schedule of $\mathcal{S} = \{0, 5, 11\}$. If we let t^* denote the starting time of the first job processed in an optimal schedule, then $t^* = 0$ in some instances as shown in the previous example, and $t^* > 0$ in others, as shown by the instance $n = 3$, $p_1 = 3, p_2 = 7, p_3 = 15$ where $q = 5$ with a unique optimal schedule of $\mathcal{S} = \{2, 5, 12\}$. We summarize these results as follows:

Proposition 4.7 *For every instance of the problem $1|a_j = q < \vartheta|\sum|S_j - a_j|$, there exists at least one of the following:*

a. an optimal schedule with $t^* = 0$

b. an optimal schedule with $S_\alpha + p_\alpha = q$.

Proof: Assume that $t^* > 0$ in an optimal solution. If a job completes at q, the proof is complete. Assuming that no job completes at q, let $0 < \epsilon < \min\{q - S_\alpha, S_\alpha + p_\alpha - q\}$, and $\Delta_\mathcal{E}$(respectively, $\Delta_\mathcal{T}$) denote the change in cost from starting all jobs ϵ units of time earlier (respectively, later). Clearly, $\Delta_\mathcal{E} = -\Delta_\mathcal{T}$, therefore $\min\{\Delta_\mathcal{E}, \Delta_\mathcal{T}\} \leq 0$, and since by assumption the solution is optimal, $\min\{\Delta_\mathcal{E}, \Delta_\mathcal{T}\} \geq 0$, thus $\Delta_\mathcal{E} = \Delta_\mathcal{T} = 0$. It is easy to see that all jobs can be moved earlier until either $t^* = 0$ or $S_\alpha + p_\alpha = q$. ■

Proposition 4.7 greatly simplifies the solution of problem instances for which $t^* > 0$ because we need consider only schedules in which $S_\alpha + p_\alpha = q$. Given $\mathcal{E}'_S$ and $\mathcal{T}_S$, Proposition 4.8 states that an optimal schedule can easily be found.

Proposition 4.8 *For every instance of the problem $1|a_j = q < \vartheta|\sum|S_j - a_j|$, in each optimal schedule the jobs in $\mathcal{E}'_S$ are ordered by non-increasing processing times, and the jobs in $\mathcal{T}_S$ are ordered by non-decreasing processing times.*

Proof: The proof follows from a job interchange argument. ■

Proposition 4.8 is different than Proposition 4.5 because the definitions of $\mathcal{E}'_S$ and $\mathcal{T}_S$ exclude the job J_α if $S_\alpha + p_\alpha > q$. Let $emin = \min\{J_j \in \mathcal{E}'_S\}$, $tmin = \min\{J_j \in \mathcal{T}_S\}$, where $emin = +\infty$ if $\mathcal{E}'_S = \emptyset$ and $tmin = +\infty$ if $\mathcal{T}_S = \emptyset$. Thus $emin$ is the index of the last job completed by time q, and $tmin$ is the index of the first job started at or after q. Now, we need the following definitions:

Definition 4.1 [Hall *et al.* , 1991] *A schedule is strongly V-shaped if* $p_\alpha \leq \min\{p_{emin}, p_{tmin}\}$.

Fact 4.1 *There exist instances of the problem* $1|a_j = q < \vartheta|\sum|S_j - a_j|$ *for which no optimal solution is strongly V-shaped.*

Proof: By example. Consider the instance $n = 5$, $p_1 = 5, p_2 = 8$, $p_3 = 15$, $p_4 = 20$, $p_5 = 21$ where $q = 7$. The unique optimal solution is $\mathcal{S} = \{8, 0, 13, 28, 48\}$, thus $p_\alpha = 8 > 5 = p_{tmin}$. ■

Definition 4.2 [Hall *et al.* , 1991] *A schedule is weakly V-shaped if* $p_\alpha \leq \max\{p_{emin}, p_{tmin}\}$.

Proposition 4.9 *For every instance of the problem* $1|a_j = q < \vartheta|\sum|S_j - a_j|$, *each optimal schedule has the weakly V-shaped property.*

Proof: Let $\mathcal{E}_S$ and $\mathcal{T}_S$ define an optimal schedule in which, contrary to Proposition 4.9, $p_\alpha > \max\{p_{emin}, p_{tmin}\}$. The following three cases are shown in Figure 4.3.

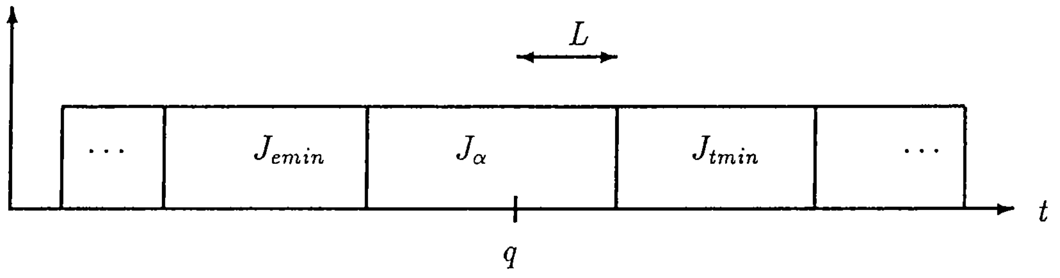


Figure 4.3: Schedule for Proposition 4.9.

1. $S_\alpha + p_\alpha = q$. Then $\alpha \in \mathcal{E}'_S$ and, from Proposition 4.8, $p_\alpha \leq p_{emin}$ in each optimal schedule, a contradiction.

2. $S_\alpha + p_\alpha = q + L$, where $p_\alpha \geq 2L$.

3. $S_\alpha + p_\alpha = q + L$, where $p_\alpha < 2L$.

The significance of the relative values of p_α and $2L$ is that if $p_\alpha \geq 2L$ we will interchange jobs J_α and J_{emin}, but if $p_\alpha < 2L$ we will interchange jobs J_α and J_{tmin}.

Let S'_j denote the starting time of job J_j after an interchange of jobs has taken place. For Case 2, let Δ denote the change in cost from interchanging jobs J_α and J_{emin}. Clearly the starting time of job J_α in the new schedule equals that of J_{emin} in Figure 4.3, so we only compute Δ with respect to the cost of scheduling the later of the two jobs. The relative values of S'_{emin} and q after the interchange provide three subcases. In each case, $\Delta < 0$ provides the necessary contradiction.

2A. $S'_{emin} < q$. Then $\Delta = (p_{emin} - L) - (p_\alpha - L) = p_{emin} - p_\alpha < 0$.

2B. $S'_{emin} = q$. Then $\Delta = 0 - (p_\alpha - L) = L - p_\alpha < 0$.

2C. $S'_{emin} > q$. Then $\Delta = (L - p_{emin}) - (p_\alpha - L) = 2L - p_\alpha - p_{emin} \leq -p_{emin} < 0$.

Consider a schedule in Case 3. Let Δ denote the change in cost from interchanging jobs J_α and J_{tmin}. Again, we need compute only the cost of scheduling the later of the two jobs. There are three subcases.

3A. $S'_\alpha > q$. Then $\Delta = [p_{tmin} - (p_\alpha - L)] - L = p_{tmin} - p_\alpha < 0$.

3B. $S'_\alpha = q$. Then $\Delta = 0 - L < 0$.

3C. $S'_\alpha < q$. Then $\Delta = [(p_\alpha - L) - p_{tmin}] - L = p_\alpha - p_{tmin} - 2L < 0$.

This completes the proof of Proposition 4.9. Thus, we know that there exists an optimal schedule with $p_\alpha \leq p_{emin}$, or $p_\alpha \leq p_{tmin}$, or both. ■

In the proofs of Propositions 4.10 and 4.11, we let Δ denote the change in the objective function value resulting from various possible adjustments to the optimal schedule.

Proposition 4.10 *If* $t^* > 0$, *then* $|\mathcal{E}_S| \geq |\mathcal{T}_S| - 3$.

Proof: If $t^* > 0$, then $S_\alpha + p_\alpha = d$. Start all jobs earlier by $0 < \epsilon < 1$ units of time. Then

$$\Delta = \epsilon\left(|\mathcal{E}_S| + 1 + 1 - (|\mathcal{T}_S| - 1)\right) \geq 0 \Rightarrow |\mathcal{E}_S| \geq |\mathcal{T}_S| - 3.$$

■

Proposition 4.11 $|\mathcal{E}_S| \leq |\mathcal{T}_S| - 1$.

Proof: Start all jobs $0 < \epsilon < 1$ units of time later. Then

$$\Delta \leq \epsilon\left(|\mathcal{T}_S| - 1 - |\mathcal{E}_S|\right) \geq 0 \Rightarrow |\mathcal{E}_S| \leq |\mathcal{T}_S| - 1.$$

■

4.1.4 $\mathcal{NP}$-Hardness of $1|a_j = q < \vartheta|\sum|S_j - a_j|$

We can state the problem $1|a_j = q < \vartheta|\sum|S_j - a_j|$ as follows:

TOTAL EARLINESS-TARDINESS WITH EQUAL-SLACK RULE:

INSTANCE: Integers y, q and n, a finite set $\mathcal{J}$ of n independent jobs, a "processing time" $p_j \in \mathcal{Z}^+$ and a "due date" $d_j \in \mathcal{Z}^+$ defined as $d_j = p_j + q$ for each $J_j \in \mathcal{J}$, $1 \leq j \leq n$.

QUESTION: Does there exist a non-preemptive schedule of these n jobs in the set $\mathcal{J}$ on one machine such that the total earliness and tardiness is no longer than y?

We use the EVEN-ODD PARTITION problem for the problem $1|a_j = q < \vartheta|\sum|S_j - a_j|$ in a similar way to Hall *et al.* (1991). In particular, we prove that answering the EVEN-ODD PARTITION problem, with data $a_1 < a_2 < \cdots < a_{2n}$, is equivalent to answering the recognition question for the problem $1|a_j = q < \vartheta|\sum|S_j - a_j|$ for the instance **P1** which has $4n+5$ jobs. The proofs of our lemmas use techniques similar to those used by Hall *et al.* (1991). We define **P1** as follows:

Instance P1

$p_{2j-1} = a_{2j-1} + B^j, \ \ j = 1, \cdots, n$

$p_{2j} = a_{2j} + B^j, \ \ j = 1, \cdots, n$

$p_{2n+1} = B^{n+1}$

$p_{2n+j} = 2q, \ \ j = 2, 3, \cdots, 2n+5$

$d = B + \sum_{j=1}^{n} B^j$

and

$$y = \sum_{j=1}^{n}(n-j+1)(p_{2j-1}+p_{2j}) + (2n+4)\sum_{j=1}^{2n+1} p_j + (2n+4)(2n+2)q$$

where $B = \sum_{j=1}^{2n} a_j/2$ and B^j denotes B raised to the power j. Observe that for this definition $p_1 \leq \cdots \leq p_{4n+5}$, provided that $B \geq 3$ and integer. We make these assumptions without loss of generality.

Lemma 4.1 *If there exists an even-odd partition, then there exists a schedule of σ for* **P1** *with $z(\sigma) = y$.*

Proof: By assumption, an even-odd partition exists. Define $b_j, \forall\ j = 1, 2, \cdots, 2n$, so that $\{b_{2j-1}, b_{2j}\} = \{a_{2j-1}, a_{2j}\}$ and $\sum_{j=1}^{n} b_{2j-1} = \sum_{j=1}^{n} b_{2j}$. The schedule of σ is illustrated in Figure 4.4. Now,

$$\begin{aligned} z(\sigma) \ &= \ 0\,(p_{2n+1}) + 1\,(p_{2n}+p_{2n-1}) + \cdots + n\,(p_2+p_1) \\ &\quad + (\sum_{j=1}^{2n+1} p_j - q) + (\sum_{j=1}^{2n+1} p_j - q + 2q) + \cdots + [\sum_{j=1}^{2n+1} p_j - q + (4n+6)\,q] = y. \end{aligned}$$

■

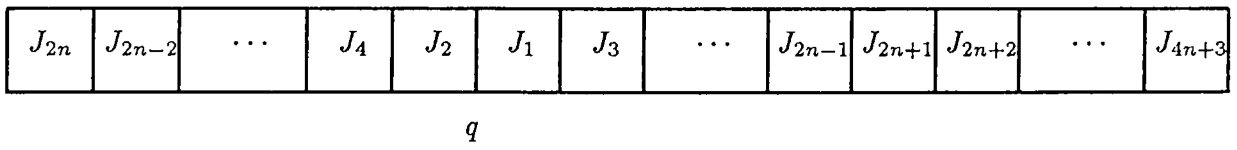


Figure 4.4: Schedule for Lemma 4.1.

Lemma 4.2 *$z(\sigma) \leq y$ for an optimal schedule of σ only if jobs $J_{2n+2}, J_{2n+3}, \cdots, J_{4n+5}$ are processed consecutively at the end of a schedule which starts at time 0.*

Proof: From Propositions 4.7 and 4.8, any optimal schedule of σ, with $t^* = 0$ in which jobs J_{2n+2}, $J_{2n+3}, \cdots$, J_{4n+5} are not scheduled last must process exactly one such job before jobs $J_1, \cdots, J_{2n+1}$. Then,

$$z(\sigma) = (4n+5)\,q + (2n+2)(2n+3)\,q + (2n+3)\sum_{j=1}^{2n+1} p_j + \sum_{j=1}^{2n}(2n+1-j)\,p_j$$

for such a schedule of σ. Thus, the difference

$$z(\sigma) - y = (2n+3)\,q - \sum_{j=1}^{2n+1} p_j + \sum_{j=1}^{n}[(n-j+1)\,p_{2j-1} + (n-j)\,p_{2j}]$$

is positive. Finally, $|\mathcal{T}_S| \geq 2n+4$ and $|\mathcal{E}_S| \leq 2n$ for each schedule in which jobs J_{2n+2},J_{2n+3},$\cdots$, J_{4n+5} are scheduled last and $t^* > 0$. Thus, by Proposition 4.9, no such schedule can be optimal. ■

From Lemma 4.2, it is obvious that it is enough to consider only the cost of the scheduling jobs $J_1, \cdots, J_{2n+1}$ consecutively, starting at time 0.

Lemma 4.3 *$z(\sigma) \leq y$ for an optimal schedule of σ only if $|\mathcal{E}_S| = n-1$, $S_\alpha + p_\alpha = q$ and job J_{2n+1} is scheduled last in σ.*

Proof: The first term in the cost of y, i.e., $0(p_{2n+1}) + 1(p_{2n} + p_{2n-1}) + \cdots + n(p_2 + p_1)$, equals the optimal cost of scheduling jobs $J_1, \cdots, J_{2n+1}$ in an unrestricted scheduling problem.

Since both $|\mathcal{E}'_S| = |\mathcal{T}'_S| + 1$ (from Proposition 4.6) and the longest job is scheduled last (from the equivalence of the problems $1|a_j = q \geq \vartheta|\sum|S_j - a_j|$ and $1|d_j = d \geq \delta|\sum|C_j - d_j|$) in any optimal schedule for the unrestricted scheduling problem with an odd number of jobs, we have $S_\alpha + p_\alpha = q$ in σ, and σ ends with job J_{2n+1}. ■

From Lemma 4.3, we know that the first n jobs scheduled must complete processing exactly at q.

Lemma 4.4 *$z(\sigma) \leq y$ for an optimal schedule of σ only if $\sum_{j\in\mathcal{E}'_S}^{2n} a_j = \sum_{j\in\mathcal{T}'_S}^{2n} a_j$, and exactly one of jobs J_{2j-1} and J_{2j} is in $\mathcal{E}'_S$ (and the other is in $\mathcal{T}'_S$), $j = 1, \cdots, n$.*

Proof: From Lemma 4.3, job J_{2n+1} must be scheduled last, and $|\mathcal{E}_S| = n - 1$. Moreover, $C_\alpha = q$. Then, since the first n jobs complete processing exactly at time q,

$$\sum_{j\in\mathcal{E}'_S}^{2n} p_j = B + \sum_{j=1}^{n} B^j = \sum_{j\in\mathcal{T}'_S}^{2n} p_j.$$

From the definitions in **P1**, this is possible only if $\mathcal{E}'_S$ (and thus $\mathcal{T}'_S$) contains exactly one of jobs J_{2j-1} and J_{2j}, $j = 1, \cdots, n$. Thus,

$$\sum_{j\in\mathcal{E}'_S}^{2n} a_j = B = \sum_{j\in\mathcal{T}'_S}^{2n} a_j,$$

and an even-odd partition exists. ■

Theorem 4.3 *The decision problem of $1|a_j = q < \vartheta|\sum|S_j - a_j|$ is $\mathcal{NP}$-complete in the ordinary sense.*

Proof: It is obvious that the decision problem of $1|a_j = q < \vartheta|\sum|S_j - a_j|$ is in $\mathcal{NP}$, and that it is possible to construct **P1** from an instance of the EVEN-ODD PARTITION problem in polynomial time. The equivalence of the two problems follows from Lemmas 4.1 - 4.4. ■

4.1.5 Concluding Remarks

When we consider Theorem 4.2, it is straightforward that we can use any algorithm developed for the problem $1|d_j = d \geq \delta|\sum|C_j - d_j|$ as a solution procedure to the problem $1|a_j = q \geq \vartheta|\sum|S_j - a_j|$. Since the former is analyzed widely in the literature, it is not fruitful to exert more effort on developing solution procedures for the latter.

The result obtained for the problem $1|a_j = q < \vartheta|\sum|S_j - a_j|$ which is stated as Theorem 4.3, states that it is most unlikely to ever develop any polynomial time algorithm for the problem $1|a_j = q < \vartheta|\sum|S_j - a_j|$. For further research, it is open to develop efficient optimizing algorithms for this problem which will be similar to those developed for the problem $1|d_j = d < \delta|\sum|C_j - d_j|$.

4.2 Problem $1|d_j|\Sigma(E_j + T_j)$ with Total-Work-Content Rule

In this part, we consider the problems in which distinct due dates are related to processing times according to Total-Work-Content rule. This means that distinct due dates are given as $d_j = kp_j$, $k > 1 \quad \forall \; j = 1, 2, \ldots, n$. Scheduling problems in which the due dates are assigned according to TWC rule drew attention of researchers who were concerned with studying the optimal due date determination. For example, Cheng (1984) examined the problem of finding the optimal processing time multiple k^* for the TWC due date assignment method and the optimal sequence to minimize the total missed due dates by minimizing the total squared value of lateness. It has been shown that k^* is a constant for a given job set and that optimal sequence should be in SPT sequence.

The intuitive explanation of TWC rule is that each job has an allowance which is directly proportional to its processing time. Hence, if it requires a larger amount of processing then it has larger time to wait in the shop to be completed on its due date. Furthermore, the target starting time of each job is also directly proportional to its processing time under TWC rule. Therefore, a job's urgency increases if its processing time is large.

In the following section, we analyze the problem of $1|d_j|\sum(E_j+T_j)$ under TWC rule and we present several properties of this problem. In Section 4.2.2, we develop a polynomial time heuristic algorithm with a very good performance.

4.2.1 Analysis of $1|d_j = k\,p_j|\sum(E_j+T_j)$

Our problem can be stated as $1|d_j = k\,p_j|\sum(E_j+T_j)$ in our notation. Since this problem permits distinct due dates that are directly proportional to the processing times, we can state several conditions under which optimal schedules are attainable easily. First of all it should be noted that all of the EDD, EST and SPT rules result in the same sequence under TWC rule. Hence, in further analyzes of the problem $1|d_j = k\,p_j|\sum(E_j+T_j)$, we will consider only EDD sequencing whenever necessary.

Throughout the analyses of the problem $1|d_j = k\,p_j|\sum(E_j+T_j)$, we could neither show that this problem is $\mathcal{NP}$-hard nor develop a polynomially bounded exact algorithm for the problem. Hence, the complexity result of the problem $1|d_j = k\,p_j|\sum(E_j+T_j)$ is still open and it is a further research topic. None the less it is possible to derive some general elimination criteria in the form of dominance properties. These elimination criteria do not necessarily produce an optimal schedule straightaway. Thus, we incorporated them into a heuristic algorithm which will be explained in Section 4.2.2. Now, we will discuss the elimination criteria in more detail. Before going further, we need the following result from the literature:

Corollary 4.1 (Rinnooy Kan, 1976) *If $z=\sum w_jT_j$ and if for two jobs J_{j_1} and J_{j_2} we have*

(i) $d_{j_1}\le d_{j_2}$,

(ii) $w_{j_1}\le w_{j_2}$, *and*

(iii) $p_{j_1}\le p_{j_2}$,

then we only have to consider schedules in which job J_{j_1} precedes job J_{j_2}.

Proof: It is obvious that the conditions (i) and (ii) imply that $T_{j_1}-T_{j_2}$ is non-decreasing on the interval [0,P]. The result follows. ■

We can use Corollary 4.1 for a scheduling problem where $z=\sum T_j$ because we have $w_j=1 \quad \forall\ j=1,2,\ldots,n$, hence condition (ii) holds for this problem also. Now, we can state the following obvious result:

Theorem 4.4 *In an optimal schedule of σ for the problem $1|d_j=k\,p_j|\sum(E_j+T_j)$, tardy jobs should be in EDD order.*

Proof: Since TWC rule is applied, for two jobs J_{j_1} and J_{j_2}, if $p_{j_1}\le p_{j_2}$ then $d_{j_1}\le d_{j_2}$. Also $w_{j_1}=w_{j_2}=1$. Hence, all conditions of Corollary 4.1 hold. The result follows. ■

Before giving the other dominance properties formally, we state the intuitive reasoning of these dominance properties. Considering the result of Theorem 4.4, we analyze the solution of the problem $1|d_j=k\,p_j|\sum(E_j+T_j)$ obtained from EDD sequencing. When we examine the structure of such a solution to the problem $1|d_j=k\,p_j|\sum(E_j+T_j)$, we see that the solution can be described by several blocks, $B_l,\quad l=1,2,\ldots,\ell$. Furthermore, it is easy to see that each block consists of three sets of jobs: the early jobs, the on-time jobs, and the tardy jobs. This can be given with our notation as $B_l=\mathcal{E}_l\cup\{J_j|C_j=d_j\}\cup\mathcal{T}_l'$. We can improve the objective function of a schedule of σ by sequencing the tardy jobs in EDD order in every block due to the Theorem 4.4. So we concentrate on the early jobs. When we further examine the structure of a solution to the problem $1|d_j = k\,p_j|\sum(E_j+T_j)$ obtained from EDD sequencing, we see that the objective function value of a schedule of σ can be improved by certain interchanges of the jobs that are early. We use the above concepts in the following dominance properties. Let job $J_\varrho \in B_l \backslash \mathcal{T}_l'$, σ defines the sequence of a schedule and σ' defines the sequence of this schedule after interchanging some of the jobs in $B_l \setminus \mathcal{T}_l'$.

Theorem 4.5 *If job J_ϱ is the last job in $B_l\backslash\mathcal{T}_l'$ and $C_\varrho \leq d_j, \quad \forall\ J_j \in B_l\backslash\mathcal{T}_l'$ in σ which is obtained from EDD sequencing, then σ' can be obtained by re-sequencing jobs in $B_l \setminus \mathcal{T}_l'$ in LPT order and $z(\sigma) > z(\sigma')$.*

Proof: If J_ϱ is the last job in $B_l \setminus \mathcal{T}_l'$ and $C_\varrho \leq d_j, \quad \forall\ J_j \in B_l \setminus \mathcal{T}_l'$ in σ which is obtained from EDD sequencing, this means that all jobs $J_j \in B_l \setminus \mathcal{T}_l'$ are early regardless of their sequencing among themselves. Hence their total earliness will be minimized by an LPT order from a discussion similar to that of the problem $1|d_j = d \geq \delta|\sum(E_j+T_j)$. ∎

Theorem 4.6 *If job J_ϱ succeeds a set of jobs $\mathcal{B}_\varrho = \{J_j \in B_l \setminus \mathcal{T}_l'\}$ and $p_j \leq p_\varrho \leq E_j$, $\forall J_j \in \mathcal{B}_\varrho$, then move job J_ϱ before jobs in $\mathcal{B}_\varrho$.*

Proof: Suppose job J_ϱ succeeds a set of jobs $\mathcal{B}_\varrho = \{J_j \in B_l \setminus \mathcal{T}_l'\}$ and $p_\varrho \leq E_j$, $\forall J_j \in \mathcal{B}_\varrho$ in the schedule of σ. Furthermore, let σ' is the sequence of the schedule obtained after moving job J_ϱ before jobs in $\mathcal{B}_\varrho$. In σ' the earliness of all jobs in $\mathcal{B}_\varrho$ will be decreased by an amount of p_ϱ (since $p_\varrho \leq E_j \quad \forall J_j \in \mathcal{B}_\varrho$) whereas the earliness of job J_ϱ increases by an amount of $\sum_{J_j \in \mathcal{B}_\varrho} p_j$. Hence

$$z(\sigma) - z(\sigma') = |\mathcal{B}_\varrho|\,p_\varrho - \sum_{J_j \in \mathcal{B}_\varrho} p_j > 0.$$

Since $z(\sigma) - z(\sigma') > 0$, it is beneficial to move job J_ϱ to the ahead of jobs in $\mathcal{B}_\varrho$. ∎

It should be noted that in Theorems 4.5 and 4.6, after an interchange or a move takes place, that is in σ', the place of jobs other than J_ϱ and $J_j \in \mathcal{B}_\varrho$ will be the same as in σ. So it is not necessary to consider those jobs in the calculation of the change of the value of the objective function. Nevertheless, it is still possible to obtain a better schedule for σ' which does not affect the results given in Theorems 4.5 and 4.6. The following theorem is a more general one and after applying this theorem, if a move takes place then a new schedule must be obtained for σ' before going further.

Theorem 4.7 *If job J_ϱ succeeds a set of jobs $\mathcal{B}_\varrho = \{J_j \in B_l \setminus \mathcal{T}_l'\}$ and $p_j \le p_\varrho$, $\forall J_j \in \mathcal{B}_\varrho$, then move job J_ϱ before jobs in $\mathcal{B}_\varrho$ if*

$$\sum_{J_j \in \mathcal{B}_\varrho} (E_j - p_j) - \sum_{J_j \in \mathcal{B}_\varrho} |E_j - p_\varrho| > 0.$$

Proof: It is easy to see from Figure 4.5 that the penalties of the jobs which will be affected from such a move are as follows: In the schedule of σ, job J_ϱ has an

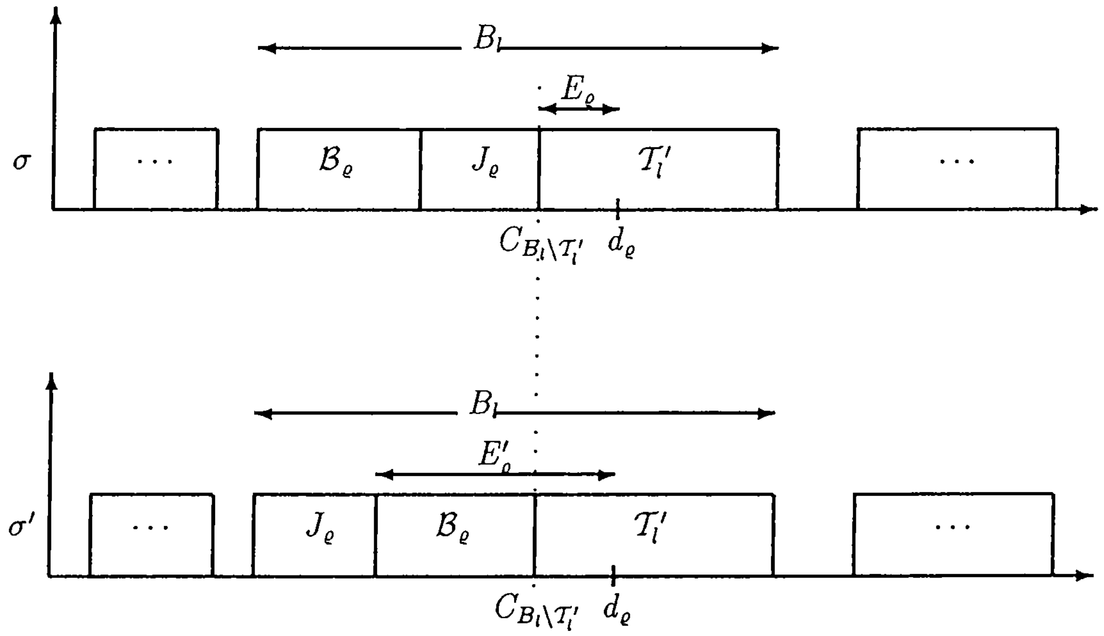


Figure 4.5: The schedules σ and σ'.

earliness of E_ϱ and all other jobs in $\mathcal{B}_\varrho$ has an earliness of E_j. After a move of job J_ϱ ahead of jobs in $\mathcal{B}_\varrho$, job J_ϱ will have an earliness of

$$E'_\varrho = E_\varrho + \sum_{J_j \in \mathcal{B}_\varrho} p_j,$$

and the jobs in $\mathcal{B}_\varrho$ will have the earliness of

$$E'_j = |E_j - p_\varrho|, \quad \forall \ J_j \in \mathcal{B}_\varrho.$$

Since no other job is affected from this move, we can write that:

$$\begin{aligned} z(\sigma) - z(\sigma') &= \sum_{J_j \in \mathcal{B}_\varrho} E_j + E_\varrho - \Big(\sum_{J_j \in \mathcal{B}_\varrho} |E_j - p_\varrho| + E_\varrho + \sum_{J_j \in \mathcal{B}_\varrho} p_j\Big) \\ &= \sum_{J_j \in \mathcal{B}_\varrho} E_j - \sum_{J_j \in \mathcal{B}_\varrho} |E_j - p_\varrho| - \sum_{J_j \in \mathcal{B}_\varrho} p_j) \\ &= \sum_{J_j \in \mathcal{B}_\varrho} (E_j - p_j) - \sum_{J_j \in \mathcal{B}_\varrho} |E_j - p_\varrho|. \end{aligned}$$

If $z(\sigma) - z(\sigma') > 0$, then this move should take place. ■

Before concluding this section, we want to give an additional property which is a special case for problem $1|d_j = k\,p_j|\sum(E_j + T_j)$.

Theorem 4.8 *If all jobs are indexed according to the* EDD *rule and* $\frac{p_{j+1}}{p_j} \geq \frac{k}{k-1}$ $\forall \ j = 1, 2, \ldots, n$, *then an* EDD *sequence gives the optimal schedule.*

Proof: The 'if' part of the theorem says that any two adjacent jobs in the EDD sequence do not overlap because the target starting time of job J_{j+1} is greater than the due date of job J_j as follows:

$$\begin{aligned} d_j &\leq a_{j+1} \\ d_j &\leq d_{j+1} - p_{j+1} \\ k\,p_j &\leq k\,p_{j+1} - p_{j+1} \\ k\,p_j &\leq (k-1)\,p_{j+1} \\ k/(k-1) &\leq p_{j+1}/p_j \end{aligned}$$

Since no two adjacent jobs overlap in EDD sequence, we can schedule all the jobs such that $C_j = d_j \;\; \forall \; j = 1, 2, \ldots, n$ which leads to an objective function value of zero which is the minimum. Hence, EDD sequence is optimal. ∎

4.2.2 A Heuristic Procedure for $1|d_j = k\,p_j|\sum(E_j + T_j)$

As it is mentioned before, since we could not be able to show neither the $\mathcal{NP}$-hardness of problem $1|d_j = k\,p_j|\sum(E_j + T_j)$ nor provide a polynomially bounded exact algorithm, we developed a very efficient heuristic procedure which incorporates the above dominance properties for $1|d_j = k\,p_j|\sum(E_j + T_j)$. This heuristic is very simple. It starts with an EDD sequence due to the Theorem 4.4 and obtain the optimal schedule for this sequence by using $OPTSCH|\sigma_{GTW}$. It then applies three other dominance properties given by Theorems 4.5, 4.6 and 4.7 in order to improve this initial schedule. A formal description of this heuristic, $HEUR\_TWC$, is as follows:

$HEUR\_TWC$:

1. Sequence the jobs in EDD order and find its optimal schedule by applying $OPTSCH|\sigma_{GTW}$.
2. Apply Theorem 4.5 and 4.6 to this initial schedule. If any interchange or move takes place, obtain the new penalties by applying $OPTSCH|\sigma_{GTW}$.
3. Apply Theorem 4.7 to every block in the last schedule obtained in the previous step. As a move takes place in a block apply $OPTSCH|\sigma_{GTW}$ to obtain the new schedule and new penalties.
4. Stop when all blocks are considered in step 3.

The first step of the algorithm is in $\mathcal{O}\ (n \log n)$ due to the sorting procedure and due to the order of the $OPTSCH|\sigma_{GTW}$ algorithm. Step 2 and Step 3 have to be applied to every block, hence the order of these steps are bounded with the number of blocks and the number of jobs, since a single application of these steps is in the order of n. That is the order of Step 3 and Step 4 is $\mathcal{O}$ (ℓn) which is dominated by $\mathcal{O}$ (n^2). Therefore, the overall complexity of the heuristic procedure *HEUR_TWC* is $\mathcal{O}$ (n^2).

In order to see the effect of the dominance properties used for the general problem in Section 3.3 on this problem, we incorporate these dominance properties into our algorithm later and tested their effectiveness.

4.2.3 Computational Experience for $1|d_j = k\,p_j|\sum(E_j+T_j)$

To test the performance of *HEUR_TWC*, we perform an experiment by generating random problems following the same method in Section 3.4. In brief, for each job J_j an integer processing time p_j was generated from the uniform distribution $[1, 100]$. Since due dates are determined according to the TWC rule, that is each $d_j = k\,p_j$, it is enough to generate k values for due date generation. The value of k is based on the sum of the processing times of all jobs ($P = \sum_{j=1}^{n} p_j$), the maximum and the minimum value of the processing times, and one of the parameters that determine the problem 'hardness'. This parameter is the *relative range of due dates* (R). We select the parameter R, because R determines the priority of the jobs. Having computed P, p_{max}, p_{min} and chosen R, an integer value of k is obtained from the following equation (Ow and Morton, 1989):

$$\lceil \frac{RP}{(p_{max} - p_{min})} \rceil.$$

Then, due dates d_j was determined from the rule that $d_j = k\,p_j$. In our computational experiment, we choose R to start from 0.2 and increase to 1.0 with 0.2 increments as in Section 3.4. We select six values to determine the problem size where $n = 8, 10, 12, 14, 16, 18$. The reason for not having greater n values is the capability of the package program CPLEX Mixed Integer Optimizer that we used for finding the optimal solution of the problems. From our experiences in Section 3.4, we know that CPLEX can not solve problems with $n > 18$.

As a result each problem set is defined with two parameters, namely R and n. For each value of n, five problems were generated for each of the 5 values of R, yielding 25 problems for each value of n and a total of 150 problems. The algorithm was coded in C and run on Sun Microsystems' Sun-4(SPARC) workstations. The optimal solutions for the test problems obtained from the package program CPLEX Mixed Integer Optimizer which runs on the same computer systems.

As it is stated in the previous section, we have an initial schedule obtained directly from EDD sequence and later some general dominance properties are added to the heuristic developed, $HEUR\_TWC$. In testing the performance of $HEUR\_TWC$, we want to see its improvement over the initial schedule. The results corresponding to the initial schedule are represented by EDD. Moreover, in order to differentiate the performance of $HEUR\_TWC$ alone and the performance of $HEUR\_TWC$ with general dominance properties, we present their results separately, denoted by $HEUR\_TWC$ and $HEUR\_TWC_{DP}$, respectively.

The results are summarized in Table 4.1. In this table we present both the average and maximum relative errors ($\overline{re}$ and re_{max}, respectively) together with the number of problems in which optimal solution found among those 25 random problems for every value of n by EDD, $HEUR\_TWC$ and $HEUR\_TWC_{DP}$ (denoted by # opt). The relative error for any rule is given as $(z_{RULE} - z^*)/z^*$ and the average and maximum relative errors are found out over 25 random problems for every n value.

When we examine the results, it is obvious that the performance of $HEUR\_TWC$ even without general dominance properties are excellent. On the maximum relative error, $HEUR\_TWC$ and $HEUR\_TWC_{DP}$ have the same value for every n value. Besides, over all 150 problems, the average relative error of $HEUR\_TWC_{DP}$ is only 3% better than that of $HEUR\_TWC$. The average

Table 4.1: Average and maximum relative errors together with number of optimal solutions found by different rules for different n values.

	EDD			*HEUR_TWC*			*HEUR_TWC*$_{DP}$		
n	$\overline{re}$	re_{max}	# opt	$\overline{re}$	re_{max}	# opt	$\overline{re}$	re_{max}	# opt
8	0.011	0.086	17	0.000	0.000	25	0.000	0.000	25
10	0.015	0.092	11	0.003	0.046	21	0.003	0.046	22
12	0.008	0.069	9	0.003	0.049	19	0.003	0.049	19
14	0.021	0.075	6	0.004	0.037	18	0.004	0.037	19
16	0.020	0.070	8	0.007	0.050	18	0.007	0.050	18
18	0.020	0.100	5	0.004	0.061	18	0.004	0.061	18
over 150 problems	0.016	0.100	56	0.0035	0.061	119	0.0034	0.061	121

relative error of *HEUR_TWC* is approximately 5 times better than that of *EDD*. Furthermore, *HEUR_TWC* finds the optimal solution 79.3 % over all 150 problems which is a very good result. When we incorporate the general dominance properties this increases only up to 80.7 % which is a very slight improvement. When we consider the percentage of optimal solutions found by *EDD* which is 37.3 %, the superiority of the performance of *HEUR_TWC* becomes clearer. As a result, *HEUR_TWC* has a very good performance and can be used for the problem $1|d_j = k\,p_j|\sum(E_j + T_j)$ as an approximate solution.

4.2.4 Concluding Remarks

As it can be seen from the computational experiments, a simple heuristic procedure performs very well on a special due date structure of the earliness and tardiness problems. These results seem encouraging for developing similar dominance properties for the general due dates and for applying those to the problem $1|d_j|\sum(E_j + T_j)$ to obtain an approximate solution. Such an extension is given in Chapter 5 for the problem $1|d_j|\sum(E_j+T_j)$. Moreover, it is worthwhile to show whether the problem $1|d_j = k\,p_j|\sum(E_j + T_j)$ is polynomially solvable or it is $\mathcal{NP}$-hard which is a further research question.

Chapter 5

Further Results for $1|d_j|\Sigma(E_j+T_j)$

In this chapter, we deal with a heuristic algorithm for problem $1|d_j|\sum(E_j+T_j)$ which is an alternative to the heuristic algorithm presented in Section 3.3. The motivation behind this algorithm is the encouraging results obtained in Section 4.2 for problem $1|d_j = k\,p_j|\sum(E_j+T_j)$. In Section 5.1, we present a new dominance property for problem $1|d_j|\sum(E_j+T_j)$ and the developed heuristic algorithm is based mainly on this simple dominance property. We also compare the performance of this simple heuristic algorithm with that of the one given in Section 3.3.

In Section 5.2, we focus on the lower bound development for $1|d_j|\sum(E_j+T_j)$ problem. We present a lower bound procedure and compare its effectiveness with one of the lower bounds from the literature on the problems that are solved to the optimality in Section 3.4.

5.1 An Alternative Heuristic Algorithm for $1|d_j|\Sigma(E_j+T_j)$

Considering the results of the analysis of problem $1|d_j = k\,p_j|\sum(E_j+T_j)$ in Section 4.2.1, we examine the applicability of dominance properties developed for problem $1|d_j = k\,p_j|\sum(E_j+T_j)$, for problem $1|d_j|\sum(E_j+T_j)$. It is easy to see that the concept of blocks can also be used for problem $1|d_j|\sum(E_j+T_j)$. Furthermore, since the most general dominance property given in Theorem 4.7 does not contain any restriction due do the structure of the due dates, we can use it for problem $1|d_j|\sum(E_j+T_j)$ as well. But it should be noted that this dominance property deals with only early jobs in a block. So, we develop another dominance property for problem $1|d_j|\sum(E_j+T_j)$ similar to Theorem 4.7 for the tardy jobs in a block as follows:

Theorem 5.1 *If job J_ϱ precedes a set of jobs $\mathcal{A}_\varrho = \{J_j \in \mathcal{T}_l'\}$ and $p_j \le p_\varrho$, $\forall J_j \in \mathcal{A}_\varrho$, then move job J_ϱ after jobs in $\mathcal{A}_\varrho$ if*

$$\sum_{J_j\in\mathcal{A}_\varrho}(T_j - p_j) - \sum_{J_j\in\mathcal{A}_\varrho}|T_j - p_\varrho| > 0.$$

Proof: This is the mirror image of Theorem 4.7 and it is easy to see from Figure 5.1 that the penalties of the jobs which will be affected from such a move are as follows: In the schedule of σ, job J_ϱ has a tardiness of T_ϱ and all other jobs in $\mathcal{A}_\varrho$ has a tardiness of T_j. After a move of job J_ϱ behind jobs in $\mathcal{A}_\varrho$, job J_ϱ will have a tardiness of

$$T_\varrho' = T_\varrho + \sum_{J_j\in\mathcal{A}_\varrho} p_j,$$

and the jobs in $\mathcal{A}_\varrho$ will have the tardiness of

$$T_j' = |T_j - p_\varrho|, \quad \forall\ J_j \in \mathcal{A}_\varrho.$$

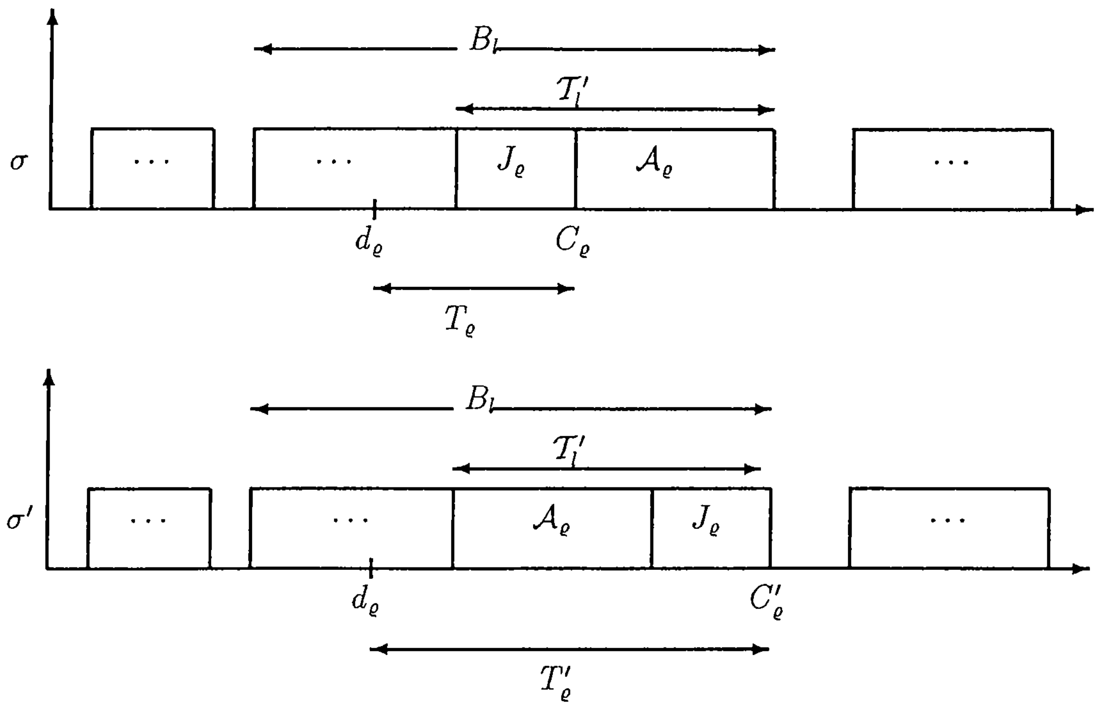


Figure 5.1: The schedules σ and σ'.

Since no other job is affected from this move, we can write that:

$$\begin{aligned} z(\sigma) - z(\sigma') &= \sum_{J_j \in \mathcal{A}_\varrho} T_j + T_\varrho - \Big(\sum_{J_j \in \mathcal{A}_\varrho} |T_j - p_\varrho| + T_\varrho + \sum_{J_j \in \mathcal{A}_\varrho} p_j\Big) \\ &= \sum_{J_j \in \mathcal{A}_\varrho} T_j - \sum_{J_j \in \mathcal{A}_\varrho} |T_j - p_\varrho| - \sum_{J_j \in \mathcal{A}_\varrho} p_j) \\ &= \sum_{J_j \in \mathcal{A}_\varrho} (T_j - p_j) - \sum_{J_j \in \mathcal{A}_\varrho} |T_j - p_\varrho|. \end{aligned}$$

If $z(\sigma) - z(\sigma') > 0$, then this move should take place. ■

We further considered the dominance properties used in the previous heuristic, *HEUR*. Since they improved the performance of *HEUR*, we incorporate those into this new heuristic, called *HEUR_ALT*, also. Hence, *HEUR_ALT* comes out to be the application of seven very simple dominance properties for problem $1|d_j|\sum(E_j + T_j)$. The following is the brief statement of *HEUR_ALT*:

HEUR_ALT:

1. Sequence the jobs in EDD order and find its optimal schedule by applying $OPTSCH|\sigma_{GTW}$.

2. Apply Theorem 4.7 and 5.1 to every block in this initial schedule. As a move takes place in a block, apply $OPTSCH|\sigma_{GTW}$ to obtain the new schedule and new penalties.

3. Apply Proposition 2.8, 3.1, 3.2, 3.3, and 3.4 to the schedule obtained in the previous step and stop.

The complexity of *HEUR_ALT* is dominated with the complexity of step 2 which is in $\mathcal{O}\ (n^2)$. Hence, the overall time complexity of *HEUR_ALT* is $\mathcal{O}\ (n^2)$.

To test the performance of *HEUR_ALT*, we use the random problems generated in Section 3.4 but only those with $n = 8, 10, 12, 14, 16, 18$ since only their optimal values can be obtained. The algorithm was coded in C and run on Sun Microsystems' Sun-4(SPARC) workstations. The optimal solutions for the test problems obtained from the package program CPLEX Mixed Integer Optimizer which runs on the same computer systems.

Since we apply some dominance properties on an initial schedule which is obtained from application of EDD rule, we want to see the improvements over the initial sequence, also. Hence, we first compare the performance of *HEUR_ALT* with that of only applying EDD rule. Then we compare the performance of *HEUR_ALT* with that of *HEUR* presented in Section 3.3.

The results of the experiment on the performance of *HEUR_ALT* is given in Table 5.1. In this table we present both the average and maximum relative errors ($\overline{re}$ and re_{max}, respectively) together with the number of problems in which optimal solution found among those 125 random problems for every value of n by *EDD*, *HEUR_ALT* and *HEUR* (denoted by # opt). The relative error for any rule is given as $(z_{RULE}-z^*)/z^*$ and the average and maximum relative errors are found for over 125 random problems for every n value.

Table 5.1: Average and maximum relative errors together with percentage of optimal solutions found by different rules for different n values.

	EDD			*HEUR_ALT*			*HEUR*		
n	$\overline{re}$	re_{max}	# opt	$\overline{re}$	re_{max}	# opt	$\overline{re}$	re_{max}	# opt
8	0.289	0.963	8	0.090	0.902	36	0.043	0.620	72
10	0.336	1.329	2	0.109	0.850	20	0.067	0.383	44
12	0.388	1.165	0	0.156	1.005	14	0.083	0.619	36
14	0.425	1.263	1	0.183	1.028	6	0.100	0.689	21
16	0.441	1.129	0	0.209	1.129	4	0.104	0.363	13
18	0.470	1.571	0	0.223	1.386	0	0.118	0.444	8
over 750 problems	0.392	1.571	11	0.162	1.386	80	0.086	0.689	194

When we examine the results, it is obvious that the performance of *HEUR_ALT* is superior to *EDD* because the average relative error of *HEUR_ALT* is 0.162 which is approximately 2.5 times better than that of *EDD*, 0.392. Moreover, while *HEUR_ALT* finds the optimal solution in 80 problems, *EDD* finds the optimal solution in only 11 problems over 750 problems. Hence, we can say that the simple dominance properties behave well for problem $1|d_j|\sum(E_j+T_j)$. Further analyses of the results reveal that the performance of *HEUR* is approximately 2 times better than that of *HEUR_ALT* when average relative errors are considered. In addition, *HEUR* finds the optimal solution in 194 problems over 750 problems. We can conclude that although simple dominance properties behave well on problem $1|d_j|\sum(E_j+T_j)$, the heuristic algorithm which is based on the incomplete dynamic programming outperforms them and application of simple rules alone leads to poorer results for problem $1|d_j|\sum(E_j+T_j)$. If the efficiency is the main concern in the solution, then a more complex heuristic algorithm like *HEUR* should be considered although it is a pseudo-polynomial algorithm.

5.2 A Lower Bound for $1|d_j|\Sigma(E_j+T_j)$

In this section, we present a lower bound procedure for problem $1|d_j|\sum(E_j+T_j)$. In obtaining a tight lower bound, we partition the job set $\mathcal{J}$ into subsets. The success of this strategy depends on the partitioning of the jobs. The jobs in a subset should be conflicting, that is they should overlap when completed at their due date. If they are not, then we get a lower bound zero. In this sense, we prefer subsets such that the executions of the jobs in the same subset interfere with each other, but not with the execution of the jobs in the other subsets. We propose a partitioning strategy that pursue this effect. This partitioning strategy uses the concepts of blocks (see Section 2.1) and is motivated by the structure of any optimal schedule. The jobs that are consecutively processed between two periods of idle time interfere with each other, but not with the other jobs. Such a partitioning is hard to obtain. To mimic such a partitioning, we identify blocks. To remind this concept, a block is a set of jobs such that for each job J_{j_1} in the block there is another job J_{j_2} in the block such that the intervals $[a_{j_1}, d_{j_1}]$ and $[a_{j_2}, d_{j_2}]$ overlap; hence, for each job in the block there exists a conflict with at least one other job in the block. However, these blocks may interfere with each other in any optimal schedule.

Let $overlap_{j_1j_2}$ denotes the length of time period that jobs J_{j_1} and J_{j_2} overlap when both jobs are scheduled at their due dates. Let σ_{EDD} and σ_{EST} denote sequences for $\mathcal{J}$ obtained by applying EDD and EST rules, respectively. We can obtain the blocks for these two sequences from the definition of block concept and assume we have ℓ blocks. From now on we talk about a single block, B_l. All of the following discussions apply to all blocks and the overall lower bound is obtained by simply summing the lower bounds of the all blocks.

Let us have the job $J_{\sigma_x(i)}$, $x =$ EDD *or* EST in block B_l. So, we determine next job that it will overlap among $J_{\sigma_{EST}(i+1)}$ and $J_{\sigma_{EDD}(i+1)}$ again in the block B_l, by applying the following:

$$\min\{overlap_{\sigma_x(i)\sigma_{EST}(i+1)}, overlap_{\sigma_x(i)\sigma_{EDD}(i+1)}\} \quad \forall\ i=1,2,\ldots,n. \tag{5.1}$$

So, for job $J_{\sigma_x(i)}$, $x=$ EDD *or* EST and $\forall\ i=1,2,\ldots,n$, we should determine the overlap with $J_{\sigma_y(i+1)}$, $y=$ EDD *or* EST as follows:

$$overlap_{\sigma_x(i)\sigma_y(i+1)} = \min\{d_{\sigma_x(i)} - a_{\sigma_y(i+1)}, d_{\sigma_y(i+1)} - a_{\sigma_x(i)}\}. \tag{5.2}$$

We state the procedure for calculating the lower bound as in the following theorem.

Theorem 5.2 *Let* $|B_l|$ *denote the number of jobs that belongs to block* B_l *and let* $\lambda_l = \sum_{\varsigma=1}^{l}|B_\varsigma|$. *The lower bound for block* B_l *is obtained as:*

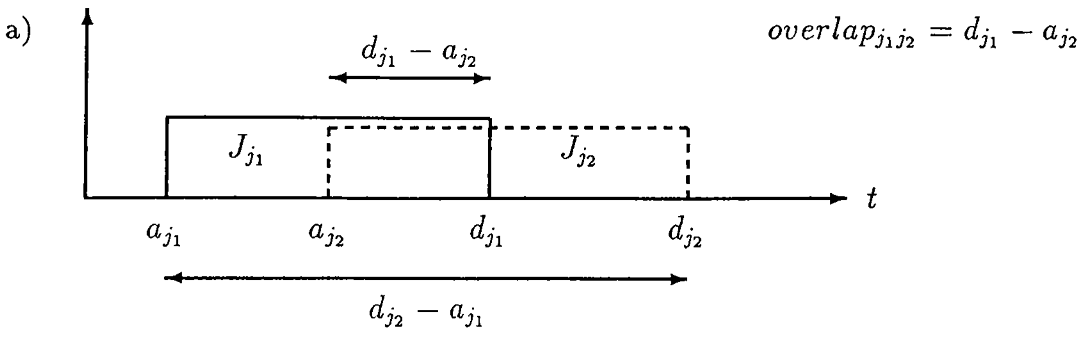
$$LB_{B_l} = \sum_{i=\lambda_{l-1}+1}^{\lambda_l-1} overlap_{\sigma_x(i)\sigma_y(i+1)}, \tag{5.3}$$

and the lower bound for the overall problem can be obtained as:

$$LB = \sum_{l=1}^{\ell} LB_{B_l}. \tag{5.4}$$

Proof: Equation 5.1 finds the job that has the minimum overlap of job $J_{\sigma_x(i)}$, $x=$ EDD *or* EST since two jobs can be closer either as to their target starting times or as to their due dates. The reasoning behind the choice of closest job to job $J_{\sigma_x(i)}$, $x=$ EDD *or* EST is to find the minimum conflicting job with job $J_{\sigma_x(i)}$, $x=$ EDD *or* EST. The logic behind this choice is as follows: If two jobs overlap, one of the jobs should be shifted either to the left or to the right of the other job to have a feasible schedule. The minimal penalty is obtained by shifting the left most job to the left (that is the job with the least target starting time is shifted to the left) or the right most job to the right (that is the job with the largest due date is shifted to the right). Once a job is considered as overlapping with another job and chosen to be shifted, it is not taken into account in further steps. That is for every job, we determine the least overlap with all other jobs.

Equation 5.2 exactly finds the overlap of two conflicting jobs. It chooses from the minimum of two periods and says that the overlap between two jobs can be the period from the due date of one job to the target starting time of the other job. The two cases are shown in Figure 5.2. It is obvious that $overlap_{j_1j_2}$ is the



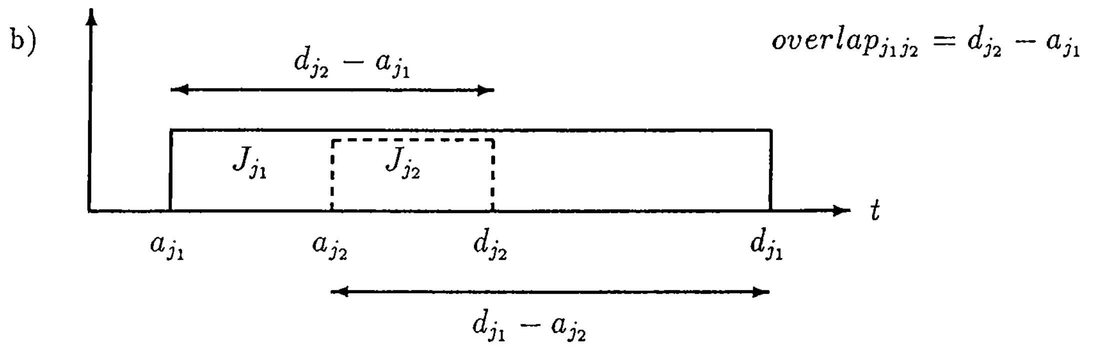


Figure 5.2: Possible overlaps for two conflicting jobs

minimal penalty that should be incurred in the objective function for any pair of jobs J_{j_1} and J_{j_2}. So a trivial lower bound for $1|d_j|\sum(E_j + T_j)$ is the summation of overlaps between the job pairs in a sequence. But the overlaps differ from one sequence to another. An apparent precedence relations between jobs can be obtained by choosing the closest job for every job as given in Equation 5.1.

After obtaining the minimum overlaps between job pairs in a block, we just simply take their summation to obtain a lower bound for that block as in Equation 5.3. It is straightforward to obtain a lower bound from the summation of the lower bounds of the blocks as in Equation 5.4 from the definition of blocks. ∎

In further analysis of this lower bound, we test its performance on the problems generated in Section 3.4. We obtain the lower bounds on the problems for $n = 8, 10, 12, 14, 16, 18$ and determine the relative error of the lower bound on the optimal solution. We then test the performance of our lower bound with another lower bound from the literature. This lower bound is due to Kim and Yano (1987) and is given as follows:

Theorem 5.3 [Kim and Yano, 1987] *If there are conflicts among two or more jobs when a set of jobs is placed as $C_j = d_j$ for all jobs J_j in the set, the total earliness and tardiness is not less than $\sum_{k\geq 2}(k-1)\,t_k$, where t_k is the length of time during which k jobs overlap.*

Proof: It is obvious that the optimal objective function value for a set of jobs is not less than the sum of the objective value of the subsets of jobs which make the set itself. We can divide a set of jobs into several subsets which are separated at times when there are no overlapping jobs. From the above statement, we only have to prove that

$$\sum_{j\in\sigma'}(E_j+T_j) \geq \sum_{k\geq 2}(k-1)\,t_k$$

in an arbitrary subset of jobs, $\sigma' = \{J_1, J_2, \ldots, J_r\}$, divided as mentioned above. We first prove that for any sequence of jobs in σ'

$$\min \sum_{j\in\sigma'}(E_j+T_j) \geq p_2 + p_3 + \cdots + p_r - (d_r - d_1).$$

Consider jobs $J_2, J_3, \ldots, J_r$ as one job J_A with processing time $p_A = \sum_{j=2}^{r} p_j$, and due date d_r. Then $\sum_{j=1}^{r}(E_j+T_j)$ is not less than $E_1 + T_1 + E_A + T_A$. This is because $E_j + T_j$, $\forall \ j = 2, 3, \ldots, r-1$, are all non-negative and $E_r + T_r = E_A + T_A$. It is obvious that for two adjacent jobs J_1 and J_A, a schedule where either $C_1 = d_1$ or $C_A = d_r$ is optimal. In both cases $E_1 + T_1 + E_A + T_A$ is not less than $\sum_{j=2}^{r} p_j - (d_r - d_1)$. Next we prove that

$$ET \equiv p_2 + p_3 + \cdots + p_r - (d_r - d_1) \geq \sum_{k=2}^{r}(k-1)\,t_k.$$

Since there is no idle time,

$$\sum_{k=1}^{r} t_k \geq p_1 + (d_r - d_1).$$

Then

$$\begin{aligned} ET &= \sum_{k=1}^{r} p_k - p_1 - (d_r - d_1) \\ &= \sum_{k=1}^{r} k\,t_k - p_1 - (d_r - d_1) \\ &= \sum_{k=2}^{r}(k-1)\,t_k + \sum_{k=1}^{r} t_k - p_1 - (d_r - d_1) \\ &\geq \sum_{k=2}^{r}(k-1)\,t_k \end{aligned}$$

since

$$\sum_{k=1}^{r} t_k \geq p_1 + (d_r - d_1).$$

The above is true for every sequence of r jobs considered. This completes the proof. ■

We test the efficiency of the proposed lower bound procedure on the random problems generated in Section 3.4 but only those with $n = 8, 10, 12, 14, 16, 18$ since only their optimal values can be obtained. The algorithm was coded in C and run on Sun Microsystems' Sun-4(SPARC) workstations. The optimal solutions for the test problems obtained from the package program CPLEX Mixed Integer Optimizer which runs on the same computer systems.

We also compare the proposed lower bound procedure, LB, with one of the lower bounds from the literature, namely with that of Kim and Yano (1987), LB_{KY}.

The results of the experiment on the performance of LB is given in Table 5.2. In this table we present both the average and maximum relative errors ($\overline{re}$ and re_{max}, respectively) together with the number of problems in which $LB > LB_{KY}$ ($\# >$). The relative error for any lower bound is given as $(z^* - z_{RULE})/z^*$ and the average and maximum relative errors are found out over 125 random problems for every n value.

Table 5.2: Average and maximum relative errors for LB and LB_{KY} and number of problems $LB > LB_{KY}$ for different n values.

	LB		LB_{KY}		
n	$\overline{re}$	re_{max}	$\overline{re}$	re_{max}	$\# >$
8	0.442	0.834	0.508	0.699	90
10	0.495	0.756	0.577	0.765	92
12	0.499	0.816	0.622	0.795	110
14	0.547	0.767	0.664	0.832	111
16	0.554	0.818	0.693	0.845	113
18	0.563	0.816	0.718	0.856	116
over 750 problems	0.517	0.834	0.630	0.856	632

When we examine the results, it is obvious that the performance of LB is superior to LB_{KY} because the average relative error of LB is always smaller than that of LB_{KY} for every value of n. Moreover, LB finds a tighter lower bound than LB_{KY} in 632 problems over 750 problems which means that in 84.3% of the problems LB outperforms LB_{KY}. This is an important aspect when such a lower bound procedure is incorporated in an implicit enumeration procedure. Hence, we can conclude that the proposed lower bound is superior to that of Kim and Yano (1987) for problem $1|d_j|\sum(E_j + T_j)$.

Chapter 6

Conclusions and Future Research

The main purpose of this study is to analyze single machine total earliness and tardiness scheduling problems with distinct due dates, namely $1|d_j|\sum(E_j + T_j)$, and to develop a dynamic programming formulation as an optimizing algorithm for its exact solution and efficient and effective heuristic solution procedures for an approximate solution. A second aspect of this study is to investigate two special structures for distinct due dates and either to show that the problem is $\mathcal{NP}$-hard even with the special structure or to propose solution procedures which bear on the characteristics of the special due date structures.

This chapter describes the contents of this study and discusses the significance and the importance of the results of this study together with possible directions for future research.

Chapter 1 defines the problem considered in this study and presents the notation used. The problem is $1|d_j|\sum(E_j + T_j)$, in which n independent jobs, each having a different due date, have to be scheduled on a single machine to minimize the total earliness and tardiness penalties. Chapter 2 identifies the characteristics of the problem $1|d_j|\sum(E_j + T_j)$ together with the different mathematical formulations. These are crucial in developing solution procedures for the problem $1|d_j|\sum(E_j + T_j)$. The literature related with the single machine earliness and tardiness scheduling problems is reviewed in Chapter 2 in a broader sense as to give an understanding of the current research on different problems related with this research. The analyses of the previous studies on the single machine earliness and tardiness scheduling problems reveal that the research on the problem $1|d_j|\sum(E_j + T_j)$ is very scarce although it constitutes an important class of problems. The research on single machine total earliness and tardiness scheduling problems mainly focused on problems with common due date structure. But, unfortunately, single machine total earliness and tardiness scheduling problem has very different characteristics regarding the different due date structures. Hence, the results obtained for problems with a common due date have a very limited applicability on problems with different due dates.

The discussion on the approaches of the problem $1|d_j|\sum(E_j + T_j)$ concludes that there is a lack of both an exact algorithm and a heuristic procedure which solves this problem efficiently and effectively. Furthermore, the work in the literature on the problem $1|d_j|\sum(E_j + T_j)$ has generally a very restrictive assumption: they do not allow idle time insertion in the schedule. Although the problem becomes easier to handle from theoretical point of view with this assumption, it does not reflect the true nature of the problem. The incorporation of earliness aspect into the objective function means to accept the possibility of idle time insertion into the schedule so long as it decreases the cost incurred. One of the contributions of this thesis is to consider the problem $1|d_j|\sum(E_j + T_j)$ without any restrictive assumptions by allowing idle time insertion whenever necessary.

One of the main chapters, Chapter 3, presents a dynamic programming formulation for the problem $1|d_j|\sum(E_j + T_j)$. This dynamic programming formulation is the first in the literature that incorporates idle time insertion in such a solution procedure. This is crucial because the incorporation of idle time into such a formulation is not so easy, since the completion times of the jobs cannot be represented simply as the sum of processing times of the jobs scheduled so far as in scheduling problems with regular performance measures. This dynamic programming formulation is extended to an incomplete dynamic programming formulation in Chapter 3.

Moreover, simple sequencing rules which are well known for classical scheduling problems are not applicable to the problem $1|d_j|\sum(E_j+T_j)$ due to their different nature. Hence, a heuristic algorithm is developed for the problem $1|d_j|\sum(E_j+T_j)$ which is based on this incomplete dynamic programming formulation and this heuristic procedure is given in Chapter 3, together with the computational results. The heuristic is pseudo-polynomial and it outperforms the heuristic given in the literature which is based on dominance properties and interchanging arguments.

Since it is difficult to obtain general and strong rules that specifies the precedence relations between jobs, two special due date structures are examined for the problem $1|d_j|\sum(E_j+T_j)$ in Chapter 4. The first structure gives an equal slack to every job and it is shown that there exist two cases for this structure in that chapter. In the first case, the optimal schedule has a nice structure and Chapter 4 first presents the equivalence of this case to a polynomially solvable problem and then proves that the other case is $\mathcal{NP}$-hard. Chapter 4 also investigates the effect of a second special structure for the due dates on the problem $1|d_j|\sum(E_j+T_j)$. That structure assigns a due date to every job as a multiple of its processing time. After giving some properties regarding to that special structure, a number of dominance properties for this problem are developed in order to obtain a sequence for the jobs. Chapter 4 concludes with an efficient heuristic algorithm.

The author know of no other work on these special due date structures in the context of the problem $1|d_j|\sum(E_j+T_j)$ although they are well studied in the job shop environments with regular performance measures. So, the results obtained in Chapter 4 contribute to the theory of scheduling.

Moreover, these results lead to a better understanding of the general problem and the first part of Chapter 5 extends the results obtained in Chapter 4 for problem $1|d_j|\sum(E_j+T_j)$ and develops an alternative heuristic algorithm for problem $1|d_j|\sum(E_j + T_j)$.

The analysis of the literature shows that there is a lack of good lower bound procedures for the problem $1|d_j|\sum(E_j + T_j)$. The second part of Chapter 5 concentrates on this issue and presents a lower bound procedure for problem $1|d_j|\sum(E_j + T_j)$. This lower bound procedure is motivated from the structure of an optimal schedule. The effectiveness of this lower bound is tested on randomly generated problems and it is shown that the developed bound is better than that of the one given in the literature.

To sum up, the four significant aspects of this study are:

(i) development of an exact dynamic programming formulation for the problem $1|d_j|\sum(E_j + T_j)$ which incorporates idle time insertion into the schedule.

(ii) development of two heuristic algorithms which behave well on the average for the problem $1|d_j|\sum(E_j + T_j)$.

(iii) analysis of two special due date structures for the problem $1|d_j|\sum(E_j + T_j)$ and to describe solution procedures for these special structures.

(iv) development of a lower bound for the problem $1|d_j|\sum(E_j + T_j)$ which behaves well compared to a lower bound from the literature.

Considering the little work on the earliness and tardiness scheduling problems on the literature, there still remains some open research questions after the contributions of this dissertation research. First, there does not exist any time-wise feasible and efficient exact algorithms specific for the problem $1|d_j|\sum(E_j + T_j)$ and the availability of such algorithms is essential for a number of reasons. To state one, it is important to have a benchmark to empirically compare the heuristic algorithms being developed, considering the capability of the current commercial package programs for mixed-integer programming models.

Considering again the exact algorithms, it will be worthwhile to develop a dynamic programming formulation which is pseudo-polynomial. Our dynamic programming formulation and the insights gained in developing the incomplete dynamic programming formulation will be a good basis for such a development. The development of a pseudo-polynomial dynamic programming formulation is important from the theory of machine scheduling point of view because this may lead to a fully-polynomial approximation scheme for earliness and tardiness scheduling problems.

Another possible research direction is to develop more efficient and effective heuristic algorithms for the problem $1|d_j|\sum(E_j + T_j)$ and to test their performance. If the exact algorithms are not time-wise feasible for large scale problems, good approximate algorithms with acceptable mean or worst case behaviors that run in real time will be meaningful for practical purposes.

In addition, the analysis of special due date structures are important in order to better understand the problem $1|d_j|\sum(E_j+T_j)$. For this reason, it is worthwhile to show whether the problem $1|d_j|\sum(E_j + T_j)$ with Total-Work-Content rule is polynomially solvable or it is $\mathcal{NP}$-hard.

Finally, the results obtained in this research can be used for total earliness and tardiness scheduling problems in multi-machine environments by either being a basis for their solution procedures or supplying approximate solutions.

We can conclude that this dissertation research will serve as a basis for answering the above further research questions.

List of Notations

$\mathcal{J}$	set of jobs
J_j	job j
n	number of jobs
p_j	processing time of job J_j
r_j	ready time of job J_j
d_j	due date of job J_j
w_j	earliness weight of job J_j
v_j	tardiness weight of job J_j
ν	$w + v$
a_j	target starting time of job J_j, $a_j = d_j - p_j$
S_j	starting time of job J_j
C_j	completion time of job J_j
E_j	earliness of job J_j
T_j	tardiness of job J_j
L_j	lateness of job J_j
$\mathcal{S}$	set of starting times
$\mathcal{C}$	set of completion times
$\mathcal{P}$	set of penalties, negative and positive values show earliness and tardiness, respectively
$g_j(E/T)$	optimality criterion which is a function of earliness and tardiness

	of each job J_j
σ, π	sequences of jobs
σ'	partial sequence of jobs
$\mathcal{J}_\sigma$	set of jobs that form the sequence σ
$\sigma(i)$	i-th job in the sequence σ
i	position index, $i = 1, 2, \ldots, n$
j	job index, $j = 1, 2, \ldots, n$
$z(\sigma)$	value of the optimality criterion for the schedule of σ
$z^*(\sigma)$	optimal value of the optimality criterion for the schedule of σ
$\mathcal{E}$	set of jobs that complete before the due date
$\|\mathcal{E}\|$	the cardinality of $\mathcal{E}$
$\mathcal{E}_S$	set of jobs that complete before the target starting time
$\|\mathcal{E}_S\|$	the cardinality of $\mathcal{E}_S$
$\mathcal{E}'$	set of jobs that complete exactly on or before the due date
$\|\mathcal{E}'\|$	the cardinality of $\mathcal{E}'$
$\mathcal{E}'_S$	set of jobs that complete exactly on or before the target starting time
$\|\mathcal{E}'_S\|$	the cardinality of $\mathcal{E}'_S$
$\mathcal{T}$	set of jobs that start exactly on or after the due date
$\|\mathcal{T}\|$	the cardinality of $\mathcal{T}$
$\mathcal{T}_S$	set of jobs that start exactly on or after the target starting time
$\|\mathcal{T}_S\|$	the cardinality of $\mathcal{T}_S$
$\mathcal{T}'$	set of jobs that complete after the due date
$\|\mathcal{T}'\|$	the cardinality of $\mathcal{T}'$
$\mathcal{T}'_S)$	set of jobs that complete after the target starting time
$\|\mathcal{T}'_S\|$	the cardinality of $\mathcal{T}'_S$
d	common due date for jobs in $\mathcal{J}$
q	common target starting time for jobs in $\mathcal{J}$
$\mathcal{P}$	class of problems for which a polynomial-bounded algorithm exists
$\mathcal{NP}$	class of problems solvable by backtrack search of polynomial-bounded depth
Y_j	idle time to be inserted before job J_j

EDD	earliest due date
EST	earliest starting time
LPT	longest processing time
SPT	shortest processing time
P	total processing time of jobs in an instance
Δ	the change in cost
δ	$\begin{cases} p_1 + p_3 + \ldots + p_n & \text{if n is odd} \\ p_2 + p_4 + \ldots + p_n & \text{if n is even} \end{cases}$
θ	$p_n + \begin{cases} p_2 + p_4 + \ldots + p_{n-1} & \text{if n is odd} \\ p_1 + p_3 + \ldots + p_{n-1} & \text{if n is even} \end{cases}$
$\mathcal{A}$	finite set of $2n$ elements, $\mathcal{A} = \{A_1, A_2, \ldots, A_{2n}\}$
a_j	"size" for every element in $\mathcal{A}$
B_l	l-th block in the schedule
$Decrease(l)$	$\{J_{\sigma(i)} \mid J_{\sigma(i)} \in B_l \vee S_{\sigma(i)} > a_{\sigma(i)}\}$
$Dec(l)$	$\lvert Decrease(l) \rvert$
$Increase(l)$	$\{J_{\sigma(i)} \mid J_{\sigma(i)} \in B_l \vee S_{\sigma(i)} \leq a_{\sigma(i)}\}$
$Inc(l)$	$\lvert Increase(l) \rvert$
$first(l)$	the smallest index of jobs in the block B_l
$last(l)$	the largest index of jobs in the block B_l
$ET_{j_1 j_2}$	minimum total earliness and tardiness of jobs J_{j_1} and J_{j_2} when J_{j_1} precedes J_{j_2}
t_{max}	$\max\{C_{j_1}, C_{j_2}\}$
LB	lower bound
UB	upper bound
$f(i,j,t)$	minimum attainable value of the penalty function for the partial schedule of $\sigma'_j \cup \{J_j\}$ when job $J_j \in \mathcal{J} \setminus \mathcal{J}_{\sigma'_j}$ is scheduled at position i and C_j being less than or equal to time t
$\mathcal{J}_{\sigma'_j}$	set of jobs scheduled before job J_j that form a partial schedule for positions from 1 to $i-1$(leading jobs of job J_j)
t	time index, $t = 1, 2, \ldots, t_{max}$
t_{max}	$\max_{J_j \in \mathcal{J}}\{d_j\} + \sum_{j=1}^{n} p_j - \min_{J_j \in \mathcal{J}}\{p_j\}$

IDP	incomplete dynamic programming
R	relative range of due dates
T	average tardiness factor
ϑ	$\begin{cases} p_1 + p_3 + \cdots + p_{n-1} & \text{if } n \text{ is even} \\ p_2 + p_4 + \cdots + p_{n-1} & \text{if } n \text{ is odd} \end{cases}$
$p_{j_1}(b)$	the amount of processing time of job J_{j_1} before q
$p_{j_1}(a)$	the amount of processing time of job J_{j_1} after q
SLK	Equal-Slack
J_α	job that starts processing before q and completes at or after q
t^*	the starting time of the first job processed in an optimal schedule
$\Delta_{\mathcal{E}}$	change in cost from starting all jobs ϵ units of time earlier
$\Delta_{\mathcal{T}}$	change in cost from starting all jobs ϵ units of time later
$emin$	$\min\{J_j \in \mathcal{E}'_S\}$
$tmin$	$\min\{J_j \in \mathcal{T}_S\}$
B	$\sum_{j=1}^{2n} a_j/2$
B^j	B raised to the power j
TWC	Total-Work-Content
k	processing time multiple in Total-Work-Content rule
$\mathcal{B}_\varrho$	$\{J_j \in B_l \setminus \mathcal{T}_l'\}$
p_{max}	maximum of processing times
p_{min}	minimum of processing times
$\overline{re}$	average relative errors
re_{max}	maximum relative errors
# opt	number of problems in which optimal solution found among generated random problems by a heuristic procedure
$\mathcal{A}_\varrho$	$\{J_j \in \mathcal{T}_l'\}$
$overlap_{j_1 j_2}$	length of time period that jobs J_{j_1} and J_{j_2} overlap when both jobs are scheduled at their due dates
λ_l	$\sum_{\varsigma=1}^{l} \lvert B_\varsigma \rvert$
LB_{B_l}	lower bound for block B_l
t_k	length of time during which k jobs overlap

References

1. Azizoğlu, M., S. Kondakçı and Ö. Kırca (1991), "Bicriteria Scheduling Problem Involving Total Tardiness and Total Earliness Penalties", *International J. Production Economics*, 23, 17–24.

2. Bagchi, U., R.S. Sullivan and Y-L. Chang (1986), "Minimizing Mean Absolute Deviation of Completion Times About a Common Due Date", *Naval Research Logistics Quarterly*, 33, 227–240.

3. Bagchi, U., Y-L. Chang and R.S. Sullivan (1987), "Minimizing Absolute and Squared Deviations of Completion Times with Different Earliness and Tardiness Penalties and a Common Due Date", *Naval Research Logistics Quarterly*, 34, 739–751.

4. Bagchi, U., R.S. Sullivan and Y-L. Chang (1987), "Minimizing Mean Squared Deviation of Completion Times About a Common Due Date", *Management Science*, 33, 894–906.

5. Baker, K.S. (1974), *Introduction to Sequencing and Scheduling*, John Wiley, New York.

6. Baker, K.S. (1984), "Sequencing Rules and Due-Date Assignments in a Job Shop", *Management Science*, 30, 1093–1104.

7. Baker, K.S. and G.D. Scudder (1990), "Sequencing with Earliness and Tardiness Penalties: A Review", *Operations Research*, 38, 22–36.

8. Bector, C.R., Y.P. Gupta and M.C. Gupta (1988), "Determination of an Optimal Common Due Date and Optimal Sequence in a Single Machine Job Shop", *International J. Production Research*, 26, 613–628.

9. Bellman, R.E. (1957), *Dynamic Programming*, Princeton University Press, Princeton.

10. Chambers, R.J., R.L. Carraway, T.J. Lowe and T.L. Morin (1991) "Dominance and Decomposition Heuristics for Single Machine Sequencing", *Operations Research*, 39, 639–647.

11. Cheng, T.C.E. (1984), "Optimal Due-Date Determination and Sequencing of n Jobs on a Single Machine", *J. Operational Research Society*, 35, 433–437.

12. Cheng, T.C.E. (1987), "An Algorithm for the CON Due-Date Determination and Sequencing Problem", *Computers and Operations Research*, 14, 537–542.

13. Cheng, T.C.E. (1988a), "An Alternative Proof of Optimality for the Common Due-Date Assignment Problem", *European J. Operational Research*, 37, 250–253.

14. Cheng, T.C.E. (1988b), "Optimal Common Due-Date with Limited Completion Time Deviation", *Computers and Operations Research*, 15, 91–96.

15. Cheng, T.C.E. (1990a), "A Note on a Partial Search Algorithm for the Single-Machine Optimal Common Due-Date Assignment and Sequencing Problem", *Computers and Operations Research*, 17, 321–324.

16. Cheng, T.C.E. (1990b), "Dynamic Programming Approach to the Single-Machine Sequencing Problem with Different Due-Dates", *Computers and Mathematical Applications*, 19, 1–7.

17. Cheng, T.C.E. and M.C. Gupta (1989), "Survey of Scheduling Research Involving Due Date Determination Decisions", *European J. Operational Research*, 38, 156–166.

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

19. Davis, J.S. and J.J. Kanet (1992), "Single-Machine Scheduling with Early and Tardy Completion Costs", *Naval Research Logistics*, to appear.

20. De, P., J.B. Ghosh and C.E. Wells (1989), "A Note on the Minimization of Mean Squared Deviation of Completion Times About a Common Due Date", *Management Science*, 35, 1143–1147.

21. Emmons, H. (1969), "One-Machine Sequencing to Minimize Certain Functions of Job Tardiness", *Operations Research*, 17, 701–715.

22. Emmons, H. (1987), "Scheduling to a Common Due Date on Parallel Uniform Processors", *Naval Research Logistics*, 34, 803–810.

23. Erlenkotter, D. (1975), "Capacity Planning for Large Multilocation Systems: Approximate and Incomplete Dynamic Programming Approaches", *Management Science*, 22, 274–285.

24. Fisher, M.L. (1976), "A Dual Algorithm for the One-Machine Scheduling Problem", *Mathematical Programming*, 11, 229–251.

25. Fry, T.D., R.D. Armstrong and J.H. Blackstone (1987), "Minimizing Weighted Absolute Deviation in Single Machine Scheduling", *IIE Transactions*, 19, 445–450.

26. Fry, T.D., G.K. Leong and T.R. Rakes (1987), "Single Machine Scheduling: A Comparison of Two Solution Procedures", *OMEGA*, 15, 277–282.

27. Fry, T.D., R.D. Armstrong and L.D. Rosen (1990), "Single Machine Scheduling to Minimize Mean Absolute Lateness: A Heuristic Solution", *Computers and Operations Research*, 17, 105–112.

28. Garey, M.R. and D.S. Johnson (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, New York.

29. Garey, M.R., R.E. Tarjan and G.T. Wilfong (1988), "One-Processor Scheduling with Symmetric Earliness and Tardiness Penalties", *Mathematics of Operations Research*, 13, 330–348.

30. Hall, N.G. (1986), "Single- and Multiple-Processor Models for Minimizing Completion Time Variance", *Naval Research Logistics Quarterly*, 33, 49–54.

31. Hall, N.G., W. Kubiak and S.P. Sethi (1991), "Earliness-Tardiness Scheduling Problems, II:Deviation of Completion Times About a Restrictive Common Due Date", *Operations Research*, 39, 847–856.

32. Hall, N.G. and M.E. Posner (1991), "Earliness-Tardiness Scheduling Problems, I:Weighted Deviation of Completion Times About a Common Due Date", *Operations Research*, 39, 836–846.

33. Hoogeveen, J.A. (1992), *Single-Machine Bicriteria Scheduling*, Unpublished Ph.D. Thesis, CWI, Amsterdam.

34. Hoogeveen, J.A. and S.L. van de Velde (1992), "Scheduling Around a Small Common Due Date", Report BS-R8914, Centre for Mathematics and Computer Science, Amsterdam.

35. Ibaraki, T. and Y. Nakamura (1990), "A Dynamic Programming Method for Single Machine Scheduling", Technical Report #90004, Kyoto University, Kyoto, Japan.

36. Kanet, J.J. (1981), "Minimizing the Average Deviation of Job Completion Times About a Common Due Date", *Naval Research Logistics Quarterly*, 28, 643-651.

37. Kim, Y-D. and C.A. Yano (1987), "Algorithms for Single Machine Scheduling Problems Minimizing Mean Tardiness and Earliness", Technical Report 87–27, The University of Michigan, Ann Arbor, Michigan.

38. Lakshminarayan, S., R. Lakshmanan, R.L. Papineau and R. Rochette (1978), "Optimal Single-Machine Scheduling with Earliness and Tardiness Penalties", *Operations Research*, 26, 1079–1082.

39. Lawler, E.L. (1977), "A Pseudopolynomial Algorithm for Sequencing Jobs to Minimize Total Tardiness", *Annals of Discrete Mathematics*, 1, 331–342.

40. Lawler, E.L., J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys (1989), "Sequencing and Scheduling: Algorithms and Complexity", Report BS-R8909, Centre for Mathematics and Computer Science, Amsterdam, The Netherlands.

41. Lawler, E.L. and J.M. Moore (1969), "A Functional Equation and its Application to Resource Allocation and Sequencing Problems", *Management Science*, 16, 77–84.

42. Morin, T.L. (1979) "Computational Advances in Dynamic Programming", *Dynamic Programming and Its Applications*, 53–90. Academic Press, New York.

43. Oğuz, C. and C. Dinçer (1992), "Single Machine Earliness-Tardiness Scheduling with Equal-Slack Rule", Research Report No.IEOR9203, Department of Industrial Engineering, Bilkent University (submitted to the *J. Operational Research Society*).

44. Oğuz, C. T.L. Morin and C. Dinçer (1992), "Incomplete Dynamic Programming for Single Machine Scheduling with both Earliness and

Tardiness Penalties", Research Report No.IEOR9210, Department of Industrial Engineering, Bilkent University.

45. Ow, P.S. and T.E. Morton (1988), "Filtered Beam Search in Scheduling", *International J. Production Research*, 26, 35–62.

46. Ow, P.S. and T.E. Morton (1989), "The Single Machine Early/Tardy Problem", *Management Science*, 35, 177–191.

47. Panwalkar, S.S., M.L. Smith and A. Seidmann (1982), "Common Due Date Assignment to Minimize Total Penalty for the One Machine Scheduling Problem", *Operations Research*, 30, 391–399.

48. Potts, C.N. and L.N. Van Wassenhove (1982) "A Decomposition Algorithm for the Single Machine Total Tardiness Problem", *Operations Research Letters*, 1, 177–181.

49. Quaddus, M. (1987), "A Generalized Model of Optimal Due-Date Assignment by Linear Programming", *J. Operational Research Society*, 38, 353–359.

50. Raghavachari, M. (1986), "A V-Shape Property of Optimal Schedule of Jobs About a Common Due Date", *European J. Operational Research*, 23, 401–402.

51. Rinnooy Kan, A.H.G. (1976), *Machine Scheduling Problems*, Martinus Nijhoff, The Hague, The Netherlands.

52. Seidmann, A., S.S. Panwalkar and M.L. Smith (1981), "Optimal Assignment of Due-Dates for a Single Processor Scheduling Problem", *International J. Production Research*, 19, 393–399.

53. Sen, T. and S.K. Gupta (1984), "A State-of-Art Survey of Static Scheduling Research Involving Due Dates", *Omega*, 12, 63–76.

54. Sidney, J.B. (1977), "Optimal Single-Machine Scheduling with Earliness and Tardiness Penalties", *Operations Research*, 25, 62–69.

55. Sundararaghavan, P.S. and M.U. Ahmed (1984), "Minimizing the Sum of Absolute Lateness in Single-Machine and Multimachine Scheduling", *Naval Research Logistics Quarterly*, 31, 325–333.

56. Szwarc, W. (1989), "Single-Machine Scheduling to Minimize Absolute Deviation of Completion Times from a Common Due date", *Naval Research Logistics*, 36, 663–673.

57. Tahboub, Z-A. A-M. and W.E. Wilhelm (1987), "Use of a Surrogate Problem to Minimize Total Earliness-Tardiness Penalties on a Single-Machine", Working Paper No 1986–010, Dept. of Industrial and Systems Engineering, The Ohio State University.

58. Yano, C.A. and Y-D. Kim (1989), "Algorithms for a Class of Single-Machine Weighted Tardiness and Earliness Problems", Technical Report, The University of Michigan, Ann Arbor, Michigan.

Vita

Ceyda Oğuz was born in Elazığ, on 3 May 1964. She attended the Department of Industrial Engineering, Middle East Technical University in September 1982 and graduated with honors in July 1986. In September 1986, she joined to the Department of Industrial Engineering, Bilkent University as a research assistant. From that time to the present, she worked with Dr. Cemal Dinçer for her graduate study at the same department. She got her M.S. degree in October 1988 with the thesis titled as *"Design and Analysis of Just-In-Time Production Systems"*. During her Ph.D. study with Dr. Cemal Dinçer, she worked on the *Single Machine Early-Tardy Scheduling Problems.* Currently, she is a research assistant at the Department of Industrial Engineering, Bilkent University.