SCHEDULING WITH DISCOUNTED REVENUES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY
AHMET KICIROĞLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF MASTER OF SCIENCE
IN
THE DEPARTMENT OF OPERATIONAL RESEARCH

SEPTEMBER, 2003

Approval of the Graduate School of Natural and Applied Sciences

Prof.Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof.Dr. Çağlar Güven
Head of Department

This is to certify that we have read this thesis and that in our opinion it is full adequate, in scope and quality, as a thesis for the degree of Master of Science.

Asst.Prof.Dr. Haldun Süral	**Prof.Dr. Meral Azizoğlu**
Co-supervisor	**Supervisor**

Examining Committee Members

Prof.Dr.Meral Azizoğlu	_____
Prof.Dr.Ömer Kırca	_____
Asst.Prof.Dr.Adil Oran	_____
Assoc.Prof.Dr.Canan Sepil	_____
Asst.Prof.Dr.Haldun Süral	_____

ABSTRACT

SCHEDULING WITH DISCOUNTED REVENUES

Kıcıroğlu, Ahmet
M.S., Department of Operational Research
Supervisor : Prof.Dr. Meral Azizoğlu
Co-supervisor : Asst. Prof.Dr. Haldun Süral
September 2003, 90 pages

Majority of the studies in the scheduling literature is devoted to time based performance measures. In this thesis, we develop a model that considers monetary issues in single machine scheduling environments. We assume all the jobs should be completed by a common due date. An early revenue is earned if the completion time is before or on the due date, and a tardy revenue is gained if the job is completed after the due date. We consider restricted and unrestricted due date versions of the problem. Our objective is the maximization of the net present value of all revenues.

We first investigate some special cases of the problem, and present polynomial time algorithms to solve them. Then, we develop branch and bound algorithms with lower and upper bounding mechanisms. Computational experiments have shown that the branch and bound algorithms can solve large-sized problems in reasonable times.

Keywords : Single machine scheduling, discounted revenues, common due date, branch and bound.

ÖZ

İSKONTO EDİLMİŞ GELİRLERLE ÇİZELGELEME

Kıcıroğlu, Ahmet
Yüksek Lisans, Yöneylem Araştırması Bölümü
Tez Yöneticisi: Prof.Dr. Meral Azizoğlu
Ortak Tez Yöneticisi : Yrd.Doç.Dr. Haldun Süral
Eylül 2003, 90 sayfa

Çizelgeleme literatüründeki çalışmaların büyük çoğunluğu zamana dayalı performans ölçütleri üzerine yoğunlaşmıştır. Bu tezde, tek makina ortamlarında parasal konuları dikkate alan bir model geliştirilmiştir. Bütün işlerin ortak bir teslim tarihinde bitirilmesi gerektiği varsayılmıştır. Bitirme zamanı teslim tarihinden önce veya tam teslim tarihinde ise erken ödeme, teslim tarihinden sonra ise geç ödeme yapılır. Hem kısıtlı teslim tarihi hem de kısıtsız teslim tarihi durumları üzerinde çalışılmıştır. Modelimizin amacı tüm gelirlerin net şimdiki değerlerinin ençoklanmasıdır.

Öncelikle, problemin bazı özel durumları belirlendi ve problemin her iki versiyonu için bu özel durumlara çözüm algoritmaları sunuldu. Daha sonra genel problem için dal-sınır algoritmaları, alt ve üst sınırlar sunuldu. Deneysel çalışma dal-sınır algoritmasının büyük boyutlu problemleri makul sürelerde çözebildiğini göstermiştir.

Anahtar Kelimeler : Tek makine çizelgeleme, iskonto edilmiş gelirler, ortak teslim tarihi, dal-sınır algoritması.

To Sena and Serra

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my supervisors Prof.Dr. Meral Azizoğlu and Asst.Prof.Dr. Haldun Süral for their perfect guidance and support throughout this study. It was an honor for me to study with them.

Special thanks to my wife Füsun for her patience and understanding, and sincere thanks to my parents for their encouragements and support. This study could not have been completed without their motivation.

I also wish to thank to my friends who shared my troubles and encouraged me to finish this study.

TABLE OF CONTENTS

LIST OF TABLES

TABLE

LIST OF FIGURES

FIGURE

CHAPTER 1

INTRODUCTION

Competition is getting harder in the market due to globalization. Raising the product quality, reducing throughput times, and increasing system flexibility become more important issues than past for a firm to be able to survive in the long run. This situation in turn has dramatically increased the perceived importance of scheduling.

Scheduling is defined as the allocation of limited resources to tasks over a specified period of time. The related decisions involve assigning the tasks to resources and finding the processing order of jobs on each resource so as to achieve the company's prespecified goal(s).

In the past, it was reasonable to keep large work-in-process and finished goods inventories to absorb scheduling errors and to meet demand fluctuation. But today, these buffers should be eliminated for a number of reasons: Rapid obsolescence of products, customer desires for shorter lead times, order revisions during that lead times, needs for more flexible reactions to shop floor problems. Manual scheduling approaches are usually inadequate and cannot provide competitive advantage to the company for the long term survival. Hence a tremendous effort has been spent for improving operational productivity through effective and efficient scheduling.

Scheduling has a broad range of application areas. The resources and tasks may take various forms. Resources may be machines in a manufacturing environment, oil tankers in an airbase, and processing units in a computer center. Tasks may be operations in a production process, refueling the aircraft for the next mission, and execution of computer programs. Employing efficient scheduling techniques helps management to achieve its organizational goals.

Setting a proper objective that fits to company's goals may not be a trivial process for several reasons: First, the important objectives like customer satisfaction due to high quality products and on time deliveries may be difficult to quantify. Second, objectives to be accomplished may be conflicting, i.e. improving one objective may cause deterioration of the other. For example, maximizing throughput, satisfying customer desires, and minimizing production costs may conflict with each other. To come up with a healthy scheduling decision, trade-offs between the objectives should be well established.

Performance measures reflect the effectiveness of the scheduling methods. In the literature, majority of the performance measures are time based. Some of these measures are makespan, total completion time and total number of tardy jobs. Makespan is equivalent to the completion time of the last job that leaves the system and the minimum makespan implies high utilization of resources. Total completion time is one of the indicators of the total holding or inventory costs incurred. Total number of tardy jobs should be kept small to increase the customer satisfaction.

Minimizing scheduling related costs such as inventory holding cost, opportunity cost, facility and labor utilization costs, and lateness cost have been rarely studied in the literature compared to the ones that consider time based criteria. When monetary issues are of concern, time value of the money becomes an important factor.

Financial status of a firm is the most important concern of a manager. They usually tend to pay more attention to monetary issues than the others. So, using monetary performance measures may lead to practical implementations that can be used by the management.

In this thesis, we have introduced a model that considers monetary issues explicitly in single machine environments. Dollar value to be earned by the completion of the jobs is the performance measure of our model. We assume all jobs should be completed by a common due date. There are two alternative revenues for each job: an early income is earned if the completion time is before or on the due date; a tardy income is gained if the job is completed after the due date. We consider two main categories according the common due date. The due date is specified in advance, i.e. restricted due date; or it is left as a decision variable which is to be solved by scheduling decisions, i.e. unrestricted due date. We try to maximize the net present values of all revenues for both due date variants. We call the problem as Earliness-Tardiness Scheduling Problem with Discounted Revenues (ETDR).

In traditional earliness-tardiness problems, earliness weights represent costs due to deterioration, need for storage and insurance, unproductive investment due to finished goods inventory; tardiness weights represent costs resulted from customer dissatisfaction, contract penalties, loss of sale and potential loss of reputation. Quantifying these costs is not an easy issue. Directly using the dollar values earned from the jobs is the advantageous side of our model, and makes it more practical. Besides this, pricing the jobs according to their early or tardy completion can be seen widely in practice. Discounting is necessary when processing times are specified as weeks or months. Hence, we believe that our model can be used easily in most of the manufacturing environments.

The rest of the thesis is organized as follows: In Chapter 2, general notation used through the text, definition and classification of the problems are presented. We also review the scheduling literature with cost based criteria, project scheduling with net present value objectives and earliness-tardiness scheduling problems in this chapter.

We analyze the unrestricted and restricted due date variants of our problem in Chapters 3 and 4, respectively. In Chapter 3, after giving a property of the optimal solution, we specify the special cases of the unrestricted due date ETDR problem and present polynomial time algorithms for these special cases. For the general problem, we present a branch and bound algorithm with lower and upper bound procedures

and discuss its computational results. Chapter 4 is devoted to the restricted due date ETDR problem. First, we give special cases and polynomial algorithms to solve these cases. Then, we present a branch and bound algorithm with upper and lower procedures and perform computational experiments to see the efficiency of the branch and bound algorithm. Finally, in Chapter 5, the discussions, conclusions and directions for future research are given.

CHAPTER 2

PROBLEM DEFINITION, CLASSIFICATION AND LITERATURE REVIEW

In this chapter, we first give the notation used through the chapters. Next, the definition and the mathematical representation of our problem and the properties of the optimal solution are presented. Then, different special cases of the problem are discussed. In the last section, we present a literature review on related scheduling problems that deal with monetary issues and earliness-tardiness weights.

2.1. NOTATION

We use standard scheduling notation throughout the text. Moreover, some new notation peculiar to our model is defined. Our basic notation is as follows:

d : Common due date

E : The set of jobs completed before or exactly on d

T : The set of jobs completed after d

N : The set of jobs where $N = E \cup T$

w_i : Weight of job i , $i \in N$

e_i : Revenue earned if $i \in E$, i.e., $w_i = e_i$, $i \in E$

t_i : Revenue earned if i∈T, i.e., $w_i = t_i,\ i \in T$

p_i : Processing time for job i, $i \in N$

α: Discount rate, $0 < \alpha \leq 1$

Given a schedule, the completion times, earliness and tardiness values of jobs are calculated as follows:

C_i : Completion time of job i for $i \in N$

E_i : Earliness of job i, $E_i=max\{0,d-C_i\}$ for $i \in N$

T_i : Tardiness of job i, $T_i=max\{0,C_i-d\}$ for $i \in N$

2.2. THE EARLINESS-TARDINESS SCHEDULING PROBLEM WITH DISCOUNTED REVENUES (ETDR)

2.2.1. Problem Definition

In this section we define the earliness-tardiness scheduling problem with discounted revenues. We assume that all the jobs should be processed without any preemption, that is, any job started processing on the machine should be kept until its completion. We consider a single machine environment. Although multi-processors are common in many production environments, the procedures for solving the single machine problems may be used as a stepping-stone for the solutions to the multi-machine case. Moreover, trying to optimize the objective(s) on individual machines, may contribute to the overall efficiency of the company. If such a machine is bottleneck, by solving the scheduling problem of this single bottleneck resource, managers may improve the efficiency of the overall system.

We assume a common due date for all jobs. Prescribing a common due date might represent a situation where several items constitute a single customer's order, or it might reflect an assembly environment in which the components should all be ready at the same time to avoid staging delays.

We assume that all the jobs are available at time zero. Revenues for each job are received once the job is completed. Depending on the completion time of the job,

one of the two different revenues are received: an early income or a tardy income. If the job is completed before or exactly at the common due date, an early income is received and if the job is completed after the common due date a tardy income is gained. Early-tardy income values are known in advance. Processing time representing the processing requirement of a job is also assumed to be deterministic.

Objective of our problem is to maximize total net present value of the revenues of the jobs, some of which are early and some of which are tardy. We assume continuous compounding to find the net present value of the total revenue. We compute the effective interest rate when the interest period and compounding period are different. We use the following equation for computing the effective interest (see Blank and Tarquin (1998)):

$$i = \left(1 + \frac{r}{m}\right)^m - 1$$

where i = effective interest rate per period; r = nominal interest rate per period; m = number of compounding periods. In case of continuous compounding, as the compounding period becomes shorter and shorter, the value of m increases, and reaches infinity. Then $i = \left(1 + \frac{r}{m}\right)^m - 1$ reduces to the following equation when $\frac{r}{m} = \frac{1}{h}$, i.e., $m = hr$ is used

$$\lim_{m\to\infty} i = \lim_{m\to\infty}\left(1 + \frac{r}{m}\right)^m - 1$$

$$= \lim_{h\to\infty}\left(1 + \frac{1}{h}\right)^{hr} - 1 = \lim_{h\to\infty}\left[\left(1 + \frac{1}{h}\right)^h\right]^r - 1$$

$$i = e^r - 1$$

We know that the present value of a dollar to be received at period n is $\frac{1}{(1+i)^n}$ where i is the interest rate. Substituting the effective interest rate, $i=e^r-1$, for continuous compounding results with e^{-rn}. The discount factor α in our model equals to term e^{-r}. Discounted cash flows usually require processing times at least weekly basis or monthly basis. As another consequence of discounting, a job may represent the group of identical subjobs which may also be considered as a lot or a batch of subjobs.

2.2.2. Mathematical Model of the ETDR

Recall that two monetary parameters of the model are the incomes earned from either early completion or tardy completion of each job. If job i is completed before or exactly at d, an early income e_i is earned at the time of job completion, otherwise a tardy income t_i is earned. As can be seen from Figure 2.1, to represent of our objective of maximizing the net present value of all early and tardy revenues, we carry all the revenues to the present time, i.e., time zero.

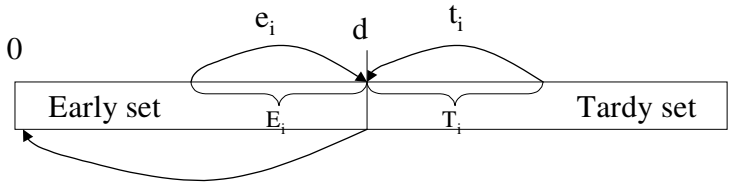

Figure 2.1. Carrying e and t values to the present time

Below is our objective function Z that maximizes the net present value given the early set E and tardy set T :

$$Z = Max(\sum_{i\in E} e_i\alpha^{-E_i} + \sum_{i\in T} t_i\alpha^{T_i})\alpha^d$$

Letting $E_i = max\{0, d\text{-}C_i\}$ and $T_i = max\{0, C_i\text{-}d\}$ we can also write the objective function as follows :

$$Z = \max(\sum_{i\in E} e_i\alpha^{-(d-C_i)} + \sum_{i\in T} t_i\alpha^{(C_i-d)})\alpha^d$$

$$\Rightarrow Z = \max(\sum_{i\in E} e_i\alpha^{C_i} + \sum_{i\in T} t_i\alpha^{C_i})\alpha^d\alpha^{-d} = \max(\sum_{i\in E} e_i\alpha^{C_i} + \sum_{i\in T} t_i\alpha^{C_i})$$

Hence, the objective function is:

$$Z = \max \sum_i w_i\alpha^{C_i} \text{ , where } w_i = e_i \text{ if } C_i \le d \text{ and } w_i = t_i \text{ if } C_i > d$$

We now give the mathematical formulation of our problem:

Decision variables:

C_{it}: completion time of job i in position t and C_{i0}=0 for all $i \in N$

$$X_{it}\begin{cases}1 & if\ \ job\ i\ is\ assigned\ to\ \ position\ t\\ 0 & otherwise\end{cases}$$

Model:

$$\text{Max} \sum_{it} w_i\alpha^{C_{it}} X_{it} \text{ , where } w_i = e_i \text{ if } C_{it} \le d \text{ and } w_i = t_i \text{ if } C_{it} > d$$

subject to

$$C_{it} \ge \sum_{j\ne i} C_{j\,t-1} + p_i - M(1 - X_{it}) \qquad \text{i=1,2,.....,n , t=1,2,.....,n} \qquad (1)$$

$$\sum_i X_{it} = 1 \qquad \text{t=1,2,.....,n} \qquad (2)$$

$$\sum_t X_{it} = 1 \qquad \text{i=1,2,.....,n} \qquad (3)$$

$$C_{it} \geq 0 \qquad \text{i=1,2,.....,n , t=1,2,.....,n} \qquad (4)$$

$$X_{it} = 0 \; or \; 1 \qquad \text{i=1,2,.....,n , t=1,2,.....,n} \qquad (5)$$

The first set of constraints provides the completion time of each job according to its position in the sequence. The second and the third constraint sets assure to assign one job to each position, and one position to each job, respectively. The last two sets are nonnegativity and binary constraints.

Note that the objective function of the model is nonlinear.

2.3. PROPERTIES OF THE OPTIMAL SOLUTIONS TO THE ETDR

As explained in the previous section, we have two alternative sets for each job: an early set or a tardy set. If the sets of the jobs are given, the problem reduces to finding the sequence of the jobs within each set so as to maximize total revenue. In this section, we show that given the job sets, the optimal ordering can be found in polynomial time.

Pinedo (1995) defines total discounted weighted completion time ($\sum w_j(1-e^{-rC_j})$) as a more general function than the conventional total weighted completion time function ($\sum w_j C_j$). He proves that in an optimal solution the jobs are ordered by nondecreasing order of $\frac{w_j e^{-rp_j}}{e^{-rp_j}-1}$ values. Below we adapt this optimal sequencing rule to our problem with discounted weights.

2.3.1. Theorem 1

In ETDR problem, an optimal sequence for each set is found by ordering the jobs within each set according to nonincreasing order of $\frac{w_i \alpha^{p_i}}{1-\alpha^{p_i}}$ ratios.

Proof:

Now, we restrict ourselves with a single set, either early or tardy set. Suppose a schedule S_1 that does not obey the rule above is optimal. Let us consider two adjacent jobs i and k having the ratios such that $\frac{w_i \alpha^{p_i}}{1-\alpha^{p_i}} < \frac{w_k \alpha^{p_k}}{1-\alpha^{p_k}}$ and job i precedes job k in S_1. Assume an adjacent pairwise interchange for these two jobs in S_1 so that job k precedes job i in the new schedule. Call this new schedule as S_2 as in Figure 2.2.

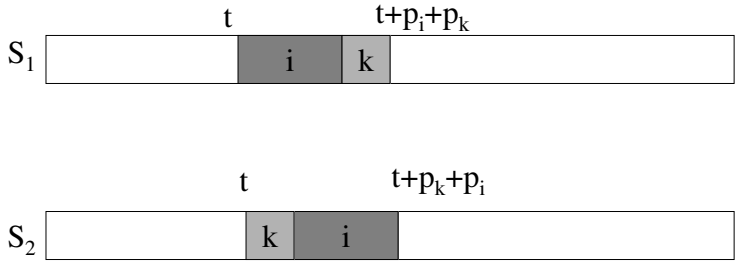

Figure 2.2. Adjacent Pairwise Interchange of jobs i and k

Observe that in schedule S_1 job i starts at t just before job k and the completion time of job k is $t + p_i + p_k$. On the other hand in schedule S_2, job k starts at time t, and job i follows job k. In this case, the completion time for job i becomes $t + p_i + p_k$ The completion times of the other jobs in the schedule, processed before or after these two jobs are not affected from this interchange. The contributions of these two jobs to the objective functions of these schedules are as follows:

$$c(S_1): w_i\alpha^{t+p_i} + w_k\alpha^{t+p_i+p_k} = \alpha^t(w_i\alpha^{p_i} + w_k\alpha^{p_i+p_k}) \text{ and}$$
$$c(S_2): w_k\alpha^{t+p_k} + w_i\alpha^{t+p_k+p_i} = \alpha^t(w_k\alpha^{p_k} + w_i\alpha^{p_k+p_i})$$

where $c(S_1)$ gives the contribution of jobs i and k to the objective function of schedule S_1.

Recall that the relationship between the ratios of these two jobs is given as follows:

$$\frac{w_i\alpha^{p_i}}{1-\alpha^{p_i}} < \frac{w_k\alpha^{p_k}}{1-\alpha^{p_k}}$$

which means that

$$w_i\alpha^{p_i}(1-\alpha^{p_k}) < w_k\alpha^{p_k}(1-\alpha^{p_i})$$

Note that $c(S_1)<c(S_2)$ as $w_i\alpha^{p_i} + w_k\alpha^{p_i+p_k} < w_k\alpha^{p_k} + w_i\alpha^{p_i+p_k}$, that is schedule S_2 has a strictly greater objective function value. This contradicts with the optimality of schedule S_1.

□

2.4. THE CLASSIFICATION OF THE ETDR PROBLEM

Early and tardy incomes, processing times, discount ratio and common due date are the parameters of our problem. Making specific changes in these parameters lead to different variants of the problem, with different solution properties and complexity statue. Here we will give the general description of some specific variants. All variants are summarized in tables in Appendix A. Solution approaches and complexities of some special cases are discussed in Chapters 3 and 4, for unrestricted and restricted cases, respectively.

2.4.1. Restricted vs Unrestricted Common Due Date

A restricted due date is specified by the customer and known by the scheduler before forming the early and tardy sets. The due date is unrestricted if it is a decision variable which should be determined simultaneously with scheduling decisions by the scheduler. Restricted and unrestricted versions of the ETDR are two main categories of our problem, and two chapters of this thesis, Chapters 3 and 4, are devoted to their analyses in detail.

2.4.2. Other Classes

Early and tardy income values may be arbitrarily different or they may be constant. Constant early and tardy incomes form a special class of our problem and the resulting problem requires an easier solution approach when compared with arbitrary e_i and t_i values. Unit processing time assumption also reduces the solution complexity.

Our decisions are related with monetary issues and the discount factor between 0 and 1 leads us to consider the time value of money. However, if the discount factor is set to 1, the effect of the discount factor on the objective function is cancelled and our problem reduces to maximizing total early and tardy revenues. This yields another special problem class.

As a result, we have two basic alternatives for each of our 5 parameters leading to 2^5 =32 different variants of the general problem. As mentioned, these 32 variants are summarized in Appendix A.

2.5. LITERATURE REVIEW

In scheduling literature, most of the studies are directed towards the time based criteria. Only few studies have considered monetary issues. In this section, we first

review the literature that takes the monetary issues into consideration. Then, we briefly review the project scheduling literature having discounted cash flows to discuss how they treat money. As it is stated before, our problem has two set definitions: early set which is composed of jobs that are completed before the common due date and tardy set with jobs that are completed after the due date. So, we will also review the traditional earliness and tardiness problems in the last section.

2.5.1. Literature on Scheduling Problems with Cost Based Approaches

At the beginning of 70s Aggarwall et al. (1973; 1974) conducted studies on cost based criteria. They stated that most of the performance criteria for scheduling problems are based on the time requirements of the jobs. However, in most of the managerial problems, the cost is a more realistic criterion than time. Scheduling rules that are oriented to optimize a time based performance measure are mainly minimizing the waiting time of high value jobs, maximizing the percentage utilization of labor or machines, minimizing the work-in-process inventory and minimizing the average lateness of jobs. If a single rule is adapted to a problem, it may cause some conflicts among the performance measures. For example, a rule that minimizes the average lateness of jobs may not maximize the machine utilization and may not minimize average work-in-process inventory. So, managers who want to optimize more than one performance criteria at a time may consider a sub optimal trade-offs by choosing one of the scheduling rules.

Aggarwal et al. (1973; 1974) tried to develop a composite scheduling rule that would optimize the overall operating costs. Four major cost components that are introduced as the operational costs are in process inventory, facilities utilization, lateness and set up time. They argued that a composite rule which is based on most relevant costs would perform at least as good as time based rules.

Dollar values of the jobs and the amount of the time a job requires to wait in the queue are taken into account for in-process inventory cost. This component is represented as below:

$$HV_i(D_{ijk}\text{-}G)$$

where

H: The cost of carrying inventory

V_i: The estimated average dollar value of the job i

D_{ijk}: Due date for the j th operation of job i in the k th machine

G: Current date

Facility utilization cost component is represented by a number inversely proportional to the processing time of a given operation as below:

$$U\left(\frac{K_{1k}}{P_{ijk}}\right)$$

where

U_k: Average hourly cost of machine utilization for the k th machine

K_{1k}: The mean processing time per operation for the k th machine

P_{ijk}: Processing time for j th operation of job i in the k th machine.

Lateness cost component is used for representing the opportunity losses due to tied-up capital and loss of future orders. They use the square of the number of the days that a job is late to increase its priority in the sequence. The term used for this component is as follows:

$$L_i\Delta\left(G - D_{ijk}\right)^2$$

where

L_i: Cost of lateness of job i per day

$\Delta(G\text{-}D_{ijk})$: Lateness of job i in number of days, $\Delta=1$ when $T>D_{ijk}$ and $\Delta=0$ otherwise.

As in the facility utilization cost, the mean set up time cost index of a given operation will be inversely proportional to its set up time and directly proportional to mean set up time of its respective machine. This term is expressed as:

$$M_k \frac{K_{2k}}{S_{ijk}}$$

where

M_k: Mean hourly set up time cost for the k th machine

K_{2k}: Mean set up time per operation for the k th machine

S_{ijk}: Set up time for the j th operation of job i on the k th machine

They combine these four cost indices into a single composite expression, and called it "Cost Rule", which is equal to

$$z_{ijk} = HV_i(D_{ijk} - G) + U_k \frac{K_{1k}}{P_{ijk}} + L_k \Delta(G - D_{ijk})^2 + M_k \frac{K_{2k}}{S_{ijk}}$$

where

z_{ijk}: The priority number (i.e. total cost index) for the j th operation of job i on the k th machine.

According to the cost rule, less priority is given to the jobs having closer due dates. Jobs that require shorter processing and set up times get higher priorities, hence incorporating the benefits of Shortest Processing Time (SPT) and Shortest Set up Time (SST) rules. Lateness factor is ineffective until it actually occurs.

They developed an experimental design to compare the efficiency of their composite cost based rule with three other well known rules on five major performance criteria. These criteria are the average total cost per job, the percentage of late jobs, the percentage of idle time, the percentage of jobs waiting in queues and the percentage of late jobs waiting in queues.

They used SPT, SST and slack per remaining operation rules (S/OPN) and their cost rule (CR) in their experiments.

They concluded that cost rule performs best on the total cost per job criterion and has the least percentage of idle time. It is the second rule that performs better after SPT on the work-in-process inventory criterion. They concluded that the cost rule is the only rule which seems to have the capacity and flexibility of satisfying most of the management objectives simultaneously. If management decides to change the emphasis from one objective to another, it can be simply done by assigning greater weight to the corresponding cost component.

Jones (1973) provided an economic framework to evaluate several dispatching rules. In this study, Jones used four different cost components: cost of idle machines, cost of carrying work-in-process inventory, cost of making excessively long promises, and penalty costs for late jobs. Idle machine costs are the sum of the out-of-pocket costs of the workers standing by the machine and opportunity costs of business forgone. It is related with the work-in-process inventory (WIP), as WIP increases, idle time for the machine decreases. Jones and Jucker (1971) pointed out that most firms in industry find that 10 % idle time for the manned machines due to lack of work is unacceptable. They choose to work with work in process inventories which cause, at most, 2 or 3 % idle time.

Jones (1973) found that the cost of carrying work-in-process inventory is a linear function of the jobs in the shop floor. Providing very long due dates may cause loss of business and to compensate the loss, an additional selling effort is needed. There are essentially no costs associated with long promises until promises get close to the length of our competitor's promises. But after that the increase in the cost due to late promises will be exponential. Another cost introduced by Jones (1973) is the cost of missed due dates. He showed the trade-offs between resource utilization and work-in- process inventory, and between length of promise and amount of tardiness.

Hoffman and Scudder (1983) investigated how the time and cost performances of the jobs are affected by the use of a rule which directly considers costs. Typical

performance measures found in the literature are the flow time, lateness and tardiness. Static rules such as shortest processing time, earliest due date, and dynamic rules which consider current slack time are used to optimize these performance measures. However, in practice, there are many other criteria used, either formally or informally, for assigning the priorities. These include cost oriented rules like processing the most profitable job first, attempting to minimize in process inventory costs.

They analyze four cost oriented rules and four time oriented rules using both time and cost based performance measures. Shortest processing time (SPT), earliest due date (EDD), minimum set up time (MNSTUP) and critical ratio (CRRAT) rules are the time oriented rules. SPT is a well known rule which selects the job having minimum processing time, including set up time. EDD selects the job having earliest due date so minimizing the maximum lateness. Job having minimum set up time is selected when MNSTUP rule is used, to maximize the machine utilization. Critical ratio rule uses the ratio slack per remaining operations.

The other four rules examined by them are directly related to the value of the job. MXPROF selects the most profitable job in the queue. DOLSHP selects the job having the highest selling price. IPDOL attempts to minimize in process costs by selecting the job with the highest current value. VALADD selects the job with the greatest value added in previous operations. The total value of a job in process is defined to be the raw material cost plus the value added.

They used six performance measures: three time based and three cost based. Lateness as average deviation from the due date, the mean job time which includes processing, set up and queue times, and average tardiness are considered as the time based criteria. Three dollar measures are average work-in-process in dollars, average profit in all queues and average value added for work waiting to be processed.

The results revealed that SPT performs best for the mean lateness and mean job time. CRRAT is best for the average tardiness and standard deviation of lateness. VALADD and IPDOL perform best for the mean work in process. MAXPROF and

DOLSHP provide good performance on the measure of average profit in the queue. They concluded that cost based rules are practical for managers when minimization of WIP and maximization of rate of return on investment are the primary and secondary objectives.

Wilson and Mardis (1983) studied the work-in-process inventory reduction. In most of the reported studies, the mean number of jobs in the shop is treated as a measure of WIP inventory. They suggested that the mean value of jobs in the shop is a more useful measure as it is a determinant of WIP inventory carrying cost. The total WIP holding cost for n jobs is $\sum_{i=1}^{n} Hv_i g_i$, where g_i is the amount of the time job i is in the shop, v_i is the dollar value of job i, and *H* is the cost of holding one dollar's worth of inventory for one time period.

Cost and time based priority scheduling is driven by profit maximization and quick response in a competitive market according to Benton (1993). He investigated two rules CR and VLADRAT which were presented before by Aggarwal et al (1973; 1974) and Scudder and Hoffman (1983) respectively. He also offered a new priority rule, the modified value added composite rule (VMOD). The VMOD rule selects the job based on the ratio between due date and profit margin.

He treated the WIP value as an increment of time spent in queue multiplied by the corresponding value added through the most recent operation completion. Another cost component he considered is the lost profit that is the sum of jobs' tardiness multiplied by their profit margin.

VLADRAT rule performs the best when mean dollar value of in process inventories is considered. However, CR and VMOD priority rules are superior against the average profit loss criterion.

Amar and Xiao (1997) provided a comprehensive cost-based formulation which is closer to the operation of an MRP-driven shop where operations are initially prioritized using backward scheduling from the promised delivery dates, and

materials and parts. They argued that if proper scheduling rules are followed at individual stations, with the aim of reducing the overall organizational cost, then collectively low overall production cost for the company will be achieved. They introduced four cost components as scheduling dependent costs associated with doing business. These costs include the following.

Holding cost: The WIP is the main component of holding costs. During periods of high interest rates, this cost could be a substantial component of a company's manufacturing cost. Any company would be interested in reducing its WIP inventory as WIP constitutes the company's working capital which neither qualifies as an investment nor as an expense. Inventory holding costs begin to be charged when raw materials ordered for the job arrives in the storage, or they are identified and reserved for any job.

Opportunity cost: This cost is due to delayed shipment of products. It is assumed that as soon as the jobs are finished, they are shipped and revenues are received. This revenue could be invested to get rewards. At the minimum this cost will be at the company's rate of return on investment. If delayed shipment would result in an additional penalty, for example loss of business, then it should be included in this component.

Inflationary spiral effect: This is a tardiness related cost component and is associated with the inflation. The inflationary spiral effect will cause an increase in the future costs. Such costs may become a concern when jobs get so much delayed in the system that the future operation costs increase.

Time value of money: By delaying the operations, we would delay adding further value to the job and thus delaying payments add time value of money on these delayed payments.

Amar and Xiao (1997) studied the objective of minimizing total accounted costs given below:

$$\sum_i \left[H(p_i + Y_i)V_i + OR_i(p_i + Y_i) + v_i e^{A_i}(e^{(f-r)Y_i} - 1) \right]$$

where

v_i = the value to be added to lot i at machine k

V_i = the value already added in lot i just before it hits machine k

p_i = processing time for lot i required at machine k

A_i = arrival time of lot i at machine k

Y_i= waiting time of lot i in line at machine k

H= holding cost per dollar per period for the company

O= company's opportunity cost per dollar per period

f = inflation rate

r = cost of money to the company

R_i = revenue that will be realized from lot i when fully completed

As a summary, relatively few studies have been conducted on the cost based scheduling approaches when compared to those performed on time based approaches. In these studies, the cost components more relevant to scheduling environment are described for providing better results to a wider range of objectives. Also, priority rules that consider these costs to optimize some financial performance measures are developed.

2.5.2. Literature on Project Scheduling

As stated by Herroelen et al. (1997) the vast majority of methodologies presented in the project scheduling literature have been developed with the objective of minimizing the project duration, and financial aspects of project management have been largely ignored. If they are not ignored, discounted cash flows generally appear in the objective of net present value (NPV) maximization.

Herroelen et al. (1997) briefly defined the NPV criterion as follows: The value of money is a function of the time of receipt or disbursement of the cash. A dollar received today is more valuable than a dollar to be received in some future time

period, because the dollar today can be invested to start earning interest immediately. To compute the NPV of a project, first an appropriate discount rate r representing the return forgone by investing in the project rather than investing in securities, should be selected. The discount factor $\beta = (1 + r)^{-1}$ denotes the present value of a dollar to be received at the end of period 1 using a discount rate r. After estimating the incremental cash flows on after tax basis, NPV of the project can be computed as:

$$NPV = F_0 + \sum_{(t)} F_t(1+r)^{-t}$$

where F_0=cash flow (usually a negative number representing the initial investment outlays) at the end of period 0 (that is today) and F_t=cash flow at the end of period *t*. Then accept-reject decision is given according to the result. The rule is to accept the project if NPV is greater than or equal to zero, and to reject if it is less than zero.

It is also important to optimize the financial returns on the basis of NPV. To accomplish this goal, contractors have historically attempted to artificially overpricing the activities to be done early in the project and under pricing those that are to be completed later which is called as front-end loading, while maintaining the overall cost of the project.

The nature and timing of the cash flows heavily depend on the contracts and on the payment structure used. The contractor would like to receive as much as possible and as early as possible while the client would like to delay payments as long as possible. A crucial distinguishing factor for contracts is whether it is fixed price or not. The most common types of fixed price contracts are bills-of-quantities contracts, price list contracts, schedule-of-rates contracts, and lump-sum contracts. Contracts which are not set for a fixed price are cost-plus percentage contracts, cost plus fixed fee contracts, target contracts, and contracts with bonuses and penalties.

The payments are subject to the conditions of the contracts. A lump sum payment may be done when the project is completed, or progress payments on the basis of time or on the basis of completion ratio may be possible. If the work is done on a

reimbursable basis (as in cost-plus contracts), after receiving invoice of the expenditure, the amount is paid to the contractor. All or partial amount may be paid in advance at the beginning of the project.

Russel (1970) studied the maximization of the net present value of cash flows when scheduling a project. His analysis considered cash out-flows associated with the expenses for initiating activities and cash in-flows that occur over the duration of the project for completing sets of activities. He showed that the cost critical path is quite different than the time critical path when monetary objectives are considered.

Smith-Daniels and Aquilano (1987) considered maximizing the net present value of a resource constrained problem. In their study, cash out-flows occur at the beginning of each activity and a single lump-sum payment (equal to the cost of activities plus a percent) is received at the completion of the project. They heuristically develop a right shifted schedule (derived from an early-start schedule by right shifting the activities subject to resource constraints) which yields a higher NPV than schedules derived with heuristics that schedule each activity as early as possible.

Yang et al. (1992) described an integer programming model for maximizing the NPV of a resource constrained problem. Cash flows occur while performing each activity. An activity requires an initial capital investment that is recovered upon completion of the activity. The value of an activity upon completion is given by

$$V_j = \sum_t (F_{jt} e^{\alpha(d_j - t)} + I_j (1 - e^{\alpha d_j}))$$

where V_j=terminal value of cash flows in activity j at its completion; F_{jt}=cash flows for activity j in period t, $t=1,2,....,d_j$; $e^{-\alpha}$=discount factor; d_j= duration of activity; I_j=capital investment required by activity j.

The first term in the equation gives the sum of the cash flows associated with an activity compounded to its end. Second term gives the opportunity cost incurred by holding the initial investment for the duration of the activity.

As a result, in project scheduling literature, we meet monetary issues when the objective is net present value maximization. Nature and timing of the cash flows throughout the project is an important issue when financial aspects are considered. We have seen that the solutions are quite different between the time based and cost based objectives.

2.5.3. Literature on Earliness – Tardiness Problems

Study of earliness and tardiness penalties in scheduling problems started to appear in the literature by 80s. Baker and Scudder (1990) consolidate these studies in their review and draw a framework of all models.

In just-in-time (JIT) scheduling environment, jobs that are completed early must be held in finished goods inventory until their due date, while jobs that are completed after their due date may cause loss of customer due to dissatisfaction. So JIT concept discourages both the early and tardy completions. Scheduling models with earliness and tardiness penalties are introduced to capture the scheduling dimensions of the JIT approach.

Scheduling research in this field generally focused on regular performance measures that are nondecreasing in job completion times. Some of these performance measures are the mean flow time, the mean lateness, and the number of tardy jobs. Efforts to cover scheduling dimensions of JIT approach lead to objectives that minimize the deviation of job completion times around the due dates. Performance criterion that measures earliness and tardiness brings a nonregular performance measure and requires different methodological issues and solution procedures than the regular ones.

Baker and Scudder (1990) described a generic earliness and tardiness model. A basic objective function for a schedule S can be written as follows:

$$f(S) = \sum_{j=1}^{n} \left[\alpha_j E_j + \beta_j T_j \right]$$

where α_j= unit earliness penalty for job j, β_j= unit tardiness penalty for job j, d_j=due date for job j, C_j=completion time of job j, $E_j = max\{0,d_j-C_j\}$ and $T_j = max\{0,C_j-d_j\}$.

In some formulations, the due date is given while in the others, the problem is to optimize the due date and job sequence simultaneously. They stated that treating due dates as decision variables reflect the practice of setting due dates internally as targets to guide the progress of shop floor activities.

Minimizing the total absolute deviation from a common due date is given as a special case in Baker and Scudder (1990). The objective function is written as

$$f(S) = \sum_{j=1}^{n} \left| C_j - d \right| = \sum_{j=1}^{n} (E_j + T_j)$$

Earliness and tardiness penalties are usually penalized at the same rate. So it is desirable to construct the schedule where the due date is in the middle of the jobs. If d is too tight, then it will not be possible to fit enough jobs in front of d. Tight d case reduces the restricted version of the problem.

An optimal solution to the unrestricted version of the problem has the following properties. There would be no inserted idle time in the schedule. Optimal schedule has a V-shaped property; i.e. jobs having $C_j \leq d$ are sequenced in nonincreasing order processing times and jobs having $C_j > d$ are sequenced in nondecreasing order of processing times. One job is completed precisely at the due date.

A restricted version of this problem was studied by Hall et al (1991). They presented a dynamic programming algorithm to find the optimal solution. They partitioned the solution space according to the optimality properties such as when schedule starts at time 0, no job completes exactly at due date; when a schedule can start after time 0 and one job completes at d.

Model with different earliness and tardiness penalties is another version of the problem. The objective function for this case is as follows:

$$f(S) = \sum_{j=1}^{n} (\alpha E_j + \beta T_j)$$

Properties for the unrestricted case of this problem were given by Baker and Scudder (1990). They showed that the optimal schedule is V-shaped with no inserted idle time. They offered an algorithm that optimally solves the problem. Hall and Posner (1991) studied this problem for job dependent penalties. They developed a dynamic programming algorithm that uses a state variable to represent the total processing time of the jobs previously scheduled as early.

The properties developed for the classical earliness-tardiness scheduling problem give us beneficial insights during our search to find optimal solution to our problem.

CHAPTER 3

UNRESTRICTED COMMON DUE DATE

In this chapter, we consider the ETDR problem where the common due date is treated as a decision variable. The manufacturer or the scheduler has to set the common due date. This leads to a situation where all possible due dates must be evaluated en route to an optimal solution. However, a property of the optimal solution (given as Theorem 2 in Section 3.1.1) defines alternative time points at which optimal common due date can be set.

The ETDR problem with arbitrary early - tardy revenues, arbitrary processing times and an arbitrary discount factor, remains open. However, we show that there are some special cases that can be solved in polynomial time. The solvable cases lead to derivation of results for the general problem, thereby helping to reduce the size of the search process.

We present a branch and bound (B&B) algorithm to obtain exact solutions. Algorithms for finding lower and upper bounds are also given. Moreover, we present a conjecture for the general problem that is motivated by the results of the computations. Results of the computational experiments with B&B algorithm are given in the last section.

3.1. A PROPERTY OF THE OPTIMAL SOLUTION

A property of the optimal solution for the unrestricted due date ETDR problem is defined in this section. This property states that there exists an optimal solution in which due date is exactly on the completion of one job.

3.1.1. Theorem 2

There exists an optimal solution to the unrestricted due date ETDR problem where one job is completed exactly at *d*.

Proof : Suppose there is an optimal schedule S in which no job is completed at d. Let job i be the job that is processed during *d*. So it belongs to tardy set. Let $0 < \varepsilon < d - (C_i - p_i)$. Sliding due date ε units until the starting time of job i, as can be seen in Figure 3.1, will not change the tardiness status of job i and its completion time. So the new schedule in which job i starts exactly at d, is as good as S.

□

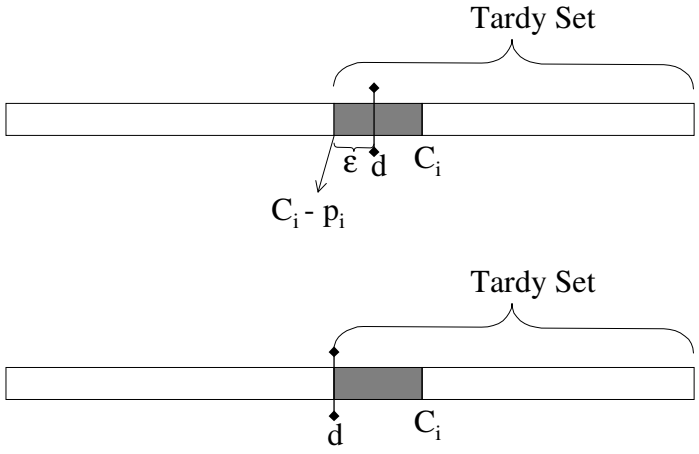

Figure 3.1 Sliding due date to the beginning of job i.

3.2. SPECIAL CASES

As explained in the previous chapter fixing some of the parameters of the ETDR problem lead to different variants of the problem. There are four parameters: early revenues, tardy revenues, processing times and discount factor for the unrestricted due date case. Each parameter having two alternatives results in 16 different cases. Here, we will discuss three basic cases: (a) discount factor $\alpha=1$, (b) processing times are unity, and (c) all early and tardy revenues are constant. Other cases are more restricted versions of these cases and are summarized in Appendix A.

3.2.1. $\alpha=1$, e_i, t_i and p_i are arbitrary

Here we assume that the discount factor is equal to 1. Early and tardy incomes of the jobs as well as processing times are arbitrary. In this case, the ETDR problem reduces to the maximization of the total job weights where job weights are defined relative to the sets they are assigned. We develop a polynomial time algorithm to solve this problem, and call it Algorithm Big Weight (BW). Algorithm BW assigns the jobs with greater e_i than t_i to the early set and assigns other jobs to the tardy set. The due date is set to the sum of processing times of the jobs assigned to set E, i.e., the completion of the last job in E. In other words, according to Algorithm BW, if $e_i > t_i$ then we set $i \in E$, if $t_i > e_i$ then we set $i \in T$. After the sets are determined, d is set equal to $\sum_{i \in E} p_i$. In equality case, i.e., $e_i=t_i$ we (arbitrarily) set $i \in T$

Theorem 3: Algorithm BW optimally solves the ETDR problem when $\alpha=1$ and due date is unrestricted.

Proof: Note that when $\alpha=1$, our objective function reduces to:

$$Z = Max\left(\sum_{i\in E} e_i \alpha^{C_i} + \sum_{i\in T} t_i \alpha^{C_i}\right) = Max\left(\sum_{i\in E} e_i 1^{C_i} + \sum_{i\in T} t_i 1^{C_i}\right) = Max\left(\sum_{i\in E} e_i + \sum_{i\in T} t_i\right).$$

Now, consider an optimal schedule S_1 in which job i with $e_i > t_i$ is in set T. Consider a new schedule S_2 where we transfer job i to set E. Let Z_{S_1} be the total revenue obtained by schedule S_1 and Z_{S_2} by schedule S_2. The difference between Z values of S1 and S_2 is $Z_{S_1} - Z_{S_2} = t_i - e_i < 0$ which contradicts with the optimality of S_1.

It can be similarly shown that in an optimal schedule, job i with $e_i < t_i$ should be processed in the tardy set. This is what Algorithm BW does.

□

The corollary below states the running time of Algorithm BW.

Corollary 1: Algorithm BW runs in O(n) time.

Proof: Each job is assigned to one set in constant time. There are n assignments, hence Algorithm BW runs in O(n) time.

□

3.2.2. $e_i=e$, $t_i=t$, p_i arbitrary, $0<\alpha<1$

In this case we assume identical early and tardy incomes for all jobs, i.e. we set $e_i=e$ and $t_i=t$ for all i. Processing times of the jobs are arbitrary and discount factor is between 0 and 1. We show that this special case can be solved in polynomial time by Algorithm Set All (SA). According to Algorithm SA, all the jobs are assigned to the early set if early income e is greater than tardy income t, otherwise we place all jobs to the tardy set. Formally, if $e > t$, we set $d = \sum p_i$ and if $e < t$ we set $d=0$. Now we have only one remaining set and we know the optimal order of the jobs from Theorem 1. Theorem 1 states that sequencing the jobs according to the nonincreasing order of $\frac{w_i \alpha^{p_i}}{1-\alpha^{p_i}}$ ratios gives the optimal sequence for each set. Here, the optimal ordering in each set reduces to Shortest Processing Time First (SPT) rule due to constant weights as shown below:

Suppose that all the jobs are assigned to early set and job i precede job k, then $\frac{e\alpha^{p_i}}{1-\alpha^{p_i}} > \frac{e\alpha^{p_k}}{1-\alpha^{p_k}}$.

This follows, $e\alpha^{p_i}\left(1-\alpha^{p_k}\right) > e\alpha^{p_k}\left(1-\alpha^{p_i}\right)$.

Since early revenue is e for all jobs, we obtain:

$\alpha^{p_i} - \alpha^{p_i+p_k} > \alpha^{p_k} - \alpha^{p_i+p_k}$.

Second terms of both sides cancel each other. Taking natural logarithm (ln) of each side of the inequality gives

$\ln \alpha^{p_i} < \ln \alpha^{p_k}$, which results in $p_i < p_k$.

This result implies that for constant weight case, an optimal ordering rule in each set reduces to the Shortest Processing Time (SPT). According to the SPT rule, we sequence the jobs in a set by nondecreasing order of their processing times.

Theorem 4: Algorithm Set All (SA) finds the optimal solution to ETDR when $e_i=e$, $t_i=t$, and the due date is unrestricted.

Proof: Consider an optimal schedule S_1 where $e>t$ and job i is in set *T*, all other jobs are in set *E*. Jobs m and q are the last two jobs of the early set. Now, we transfer job i to the early set. Assume its position in the early set be just before job k according to SPT and call this new schedule as S_2. As can be seen from Figure 3.2, when job i is transferred to *E*, jobs between tardy and early positions of job i slide backwards by p_i units. Note that the completion time of job i in S_1 and the completion time of job m in S_2 are equal.

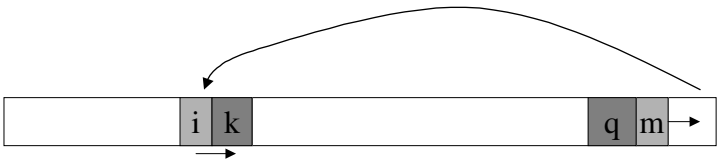

Figure 3.2 Transfer of job i from tardy set to early set

Let Z_{S_1} (Z_{S_2}) be the total revenues obtained by schedule S_1 (S_2). The difference between the Z values of S_1 and S_2 is due to changing job i from tardy to early and sliding the jobs between two positions of job i backwards by p_i units. Mathematically, we can express this difference as:

$$Z_{S_1} - Z_{S_2} = t\alpha^{C_{iT}} - e\alpha^{C_{iE}} + e\alpha^{C_k} - e\alpha^{C_k+p_i} + e\alpha^{C_l} - e\alpha^{C_l+p_i} + \ldots$$
$$+ e\alpha^{C_q} - e\alpha^{C_q+p_i} + e\alpha^{C_m} - e\alpha^{C_{iT}}$$

where

C_{iE} : Completion time of job i when $i \in E$

C_{iT} : Completion time of job i when $i \in T$

The first and second terms represent the difference caused by changing the position of job i from tardy to early. The other term pairs with positive and negative signs represent the difference caused by sliding the jobs p_i units back.

When we compare first and last terms we see that $t\alpha^{C_{iT}} - e\alpha^{C_{iT}} < 0$. Comparing second and third terms gives $-e\alpha^{C_{iE}} + e\alpha^{C_k} < 0$ as $p_i < p_k$ (or $C_{iE} < C_k$) and fourth and fifth terms also gives $-e\alpha^{C_k+p_i} + e\alpha^{C_l} < 0$ as $p_i + p_k < p_k + p_l$ (or $C_k + p_i < C_l$) This is valid for all other pairs and hence $Z_{S_1} - Z_{S_2} < 0$. This contradicts with the optimality of S_1.

Now we look to the reverse situation, i.e. we have an optimal schedule S_1 with $t>e$, job i is in the early set and all other jobs are in the tardy set. Job k is the first job of tardy set. Job i is transferred to the tardy set. Assume its order in the tardy set is after jobs q and m. Schedule S_2 is obtained.

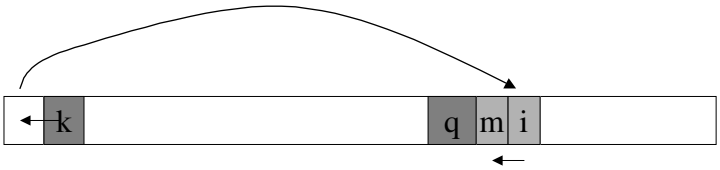

Figure 3.3 Transfer of job i from early set to tardy set

As can be seen from Figure 3.3, when job i is transferred to set *T*, jobs between tardy and early positions of job i slide forwards by p_i units. Note that the completion time of job m in schedule S_1 and the completion time of job i in schedule S_2 are equal.

Let $Z_{S_1}(Z_{S_2})$ be the total revenue obtained by schedule S_1 (S_2). The difference between Z values of S_1 and S_2 is due to changing job i from early to tardy and sliding the jobs between new and old positions of job i forwards by p_i units. Mathematically, we can express this difference as below:

$$Z_{S_1} - Z_{S_2} = e\alpha^{C_{iE}} - t\alpha^{C_{iT}} + t\alpha^{C_k+p_i} - t\alpha^{C_k} + t\alpha^{C_l+p_i} - t\alpha^{C_l} + \ldots$$
$$+ t\alpha^{C_q+p_i} - t\alpha^{C_q} + t\alpha^{C_{iT}} - t\alpha^{C_m}$$

where

C_{iE} : Completion time of job i when $i \in E$

C_{iT} : Completion time of job i when $i \in T$

The first and second terms represent the difference caused by changing job i from early to tardy. The other pairs with opposite signs represent the difference caused by sliding the jobs p_i units forwards.

$-t\alpha^{C_{iT}}$ and $t\alpha^{C_{iT}}$ terms cancel each other. Comparing first and fourth terms, we find that $e\alpha^{C_{iE}} - t\alpha^{C_k} < 0$ as $e<t$ and $p_k<p_i$. Comparing third and fifth terms also gives $t\alpha^{C_k+p_i} - t\alpha^{C_l} < 0$ as $p_k+p_l<p_k+p_i$. This holds for all other pairs and hence $Z_{S_1} - Z_{S_2} < 0$, this contradicts with the optimality of S_1.

□

The corollary below gives the complexity of the algorithm.

Corollary 2: Algorithm SA runs in O(nlogn) time.

Proof: After the sets of the jobs are determined, the problem reduces to forming the SPT sequence in O(nlogn) time.

□

3.2.3. $p_i = 1$, e_i and t_i are arbitrary, $0<\alpha<1$

When all jobs have unit processing times, the ETDR problem reduces to a linear assignment problem. Note that the job assigned to position j completes at time j as $p_i=1$ for i. Therefore, we should decide only to assign which job to which position so as to maximize total discounted revenue. As there are n possibilities for the common due date, we have to solve the assignment problem n times among these n solutions. The optimal solution is the one giving the maximum total revenue. Mathematical representation of the assignment problem is given below:

Let

$$X_{ij} = \begin{cases} 1 & if\ \ job\ i\ assigned\ \ position\ \ j \\ 0 & otherwise \end{cases}$$

The objective function of the assignment problem is

$$Max\sum_{ij} c_{ij} X_{ij}$$

where c_{ij} is the revenue of assigning job i to position j. Note that $c_{ij}=e_i\alpha^j$ if $i\in E$ and $c_{ij}=t_i\alpha^j$ if $i\in T$. Hence the objective function reduces to:

$$Max\left[\sum_{\substack{ij\\ i\in E}}^{n} X_{ij} e_i \alpha^j + \sum_{\substack{ij\\ i\in T}}^{n} X_{ij} t_i \alpha^j\right]$$

The assignment constraints are:

$$\sum_{i=1}^{n} X_{ij} = 1 \qquad \text{j = 1,2.....,n}$$

$$\sum_{j=1}^{n} X_{ij} = 1 \qquad \text{i = 1,2.....,n}$$

$$X_{ij} = 0-1 \qquad \text{for all i and j.}$$

Corollary 3 : The unrestricted due date ETDR problem is solved $O(n^4)$ steps when $p_i=1$.

Proof : The assignment algorithm can be solved in $O(n^3)$ time (see, for example Janker and Volgenant (1987)). The solution to unrestricted due date ETDR problem requires optimal solutions of assignment algorithm for each possible value of d. There are n possibilities for d. Therefore the problem is solved $O(n^4)$ steps.

□

3.3 SOLUTION APPROACHES TO UNRESTRICTED DUE DATE ETDR

In this section, we consider the ETDR problem with unrestricted due date with arbitrary parameters. We present lower bound and upper bound procedures and then give a branch and bound algorithm that solves the problem optimally.

3.3.1 Lower Bound

We use a polynomial time algorithm for finding a lower bound to our problem We call this algorithm as Polynomial Rule (PR). Algorithm PR is based on the idea that while forming the optimal solution, the job that enters the early set at a specific due date *d*, will be in the early set for all due dates greater than *d*. The algorithm starts with all jobs in tardy set, then the jobs are assigned to early set one by one temporarily. In each assignment step, the job that provides the maximum improvement in the objective function is transferred to set *E*. After the first job is determined, the second job that would enter to the early set is searched. If there is such a job, it is assigned to the early set. This procedure is repeated until no job remains in the tardy set. In each iteration, the objective function value is recorded. When the procedure ends, we select the maximum of the recorded solutions. The steps of the algorithm are as follows:

Algorithm PR

Step 1. Let $d = 0$. Set $T = \{i \mid i \in N\}$ *and* $E = \emptyset$.

Find $R_{ti} = \dfrac{t_i \alpha^{p_i}}{1-\alpha^{p_i}}$ ratio for each $i \in T$

Order all the jobs in tardy set according to nonincreasing order of R_{ti} values. Compute objective function value of this sequence.

Step 2. Calculate the objective function value of the sequence for each job when transferred from set *T* to set *E*. Select the job that gives the maximum objective function value and assign it to *E*. Let this job be r. Let Z_r be the corresponding

objective function value when job r is assigned to the early set. Set $E \leftarrow E \cup \{r\}$, $T \leftarrow T \mid \{r\}$.

Step 3. Set $d = \sum_{i \in E} p_i$

Step 4. If $T=\emptyset$ go to Step 5 else go to Step 2.

Step 5. The best solution is $Z = Max\{Z_r\}$.

Corollary 4 states the complexity of Algorithm PR.

Corollary 4 : Algorithm PR runs in $O(n^2)$ time.

Proof: The optimal orderings in the sets are formed in O(nlogn). For each possible d, algorithm selects a job in O(n) time for assigning early set. There are n possibilities for d, thereby increasing the complexity to $O(n^2+nlogn)$, hence $O(n^2)$.

□

We illustrate Algorithm PR through the following 4 jobs problem. Whose data are reported in Table 3.1. We assume α=0.9 and let R_{ei} and R_{ti} be $R_{ei} = \frac{e_i \alpha^{p_i}}{1-\alpha^{p_i}}$ and $R_{ti} = \frac{t_i \alpha^{p_i}}{1-\alpha^{p_i}}$, respectively.

Table 3.1. Data for the example problem with arbitrary processing times

job	e_i	t_i	p_i	R_{ei}	R_{ti}
i	5	3	2	21.31	12.78
j	8	5	6	9.07	5.67
k	6	7	3	16.14	18.83
m	2	4	4	3.81	7.63

Step 1. d=0 and $R_{ti} = \frac{t\alpha^{p_i}}{1-\alpha^{p_i}}$ values are computed for each job i and are given in Table 3.1. The sequence of jobs is as in Figure 3.4.

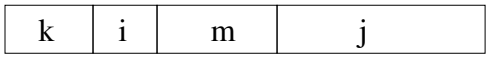

Figure 3.4. Job sequence when d=0

The objective value of the sequence is $Z_0 = t_k\alpha^{Ck} + t_i\alpha^{Ci} + t_m\alpha^{Cm} + t_j\alpha^{Cj} = 9.4536$

Step 2. Transfer each job one by one to early set as shown in Figure 3.5. and we come up with four schedules. The objective function value of each schedule is computed as follows.

$Z_k = e_k\alpha^{Ck} + t_i\alpha^{Ci} + t_m\alpha^{Cm} + t_j\alpha^{Cj} = 8.7246$

$Z_i = e_i\alpha^{Ci} + t_k\alpha^{Ck} + t_m\alpha^{Cm} + t_j\alpha^{Cj} = 10.7625$ *

$Z_m = e_m\alpha^{Cm} + t_k\alpha^{Ck} + t_i\alpha^{Ci} + t_j\alpha^{Cj} = 6.8519$

$Z_j = e_j\alpha^{Cj} + t_k\alpha^{Ck} + t_i\alpha^{Ci} + t_m\alpha^{Cm} = 8.7284$

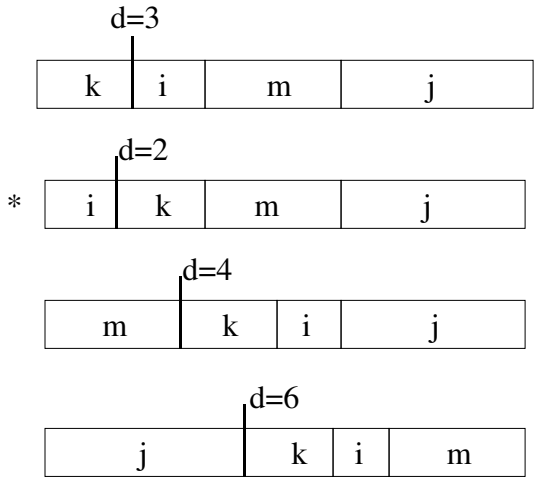

Figure 3.5. Transferring jobs T to E

We assign job i to early set since *Max* $\{Z_k, Z_i, Z_m, Z_j\} = Z_i$.

Set E={i} and T={k,m,j}.

Step 3. Set d=2 and Z_1=10.7625.

Step 4. Repeat steps 2 and 3 by transferring the second job to E. The resulting three schedules are given in Figure 3.6.

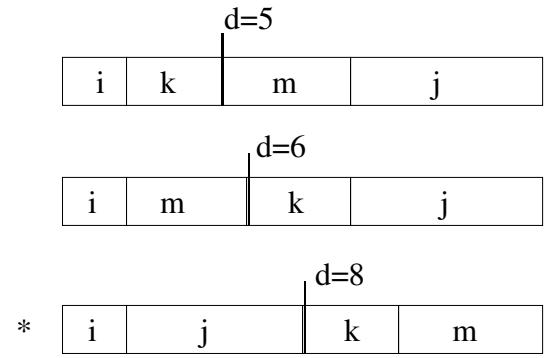

Figure 3.6. Three possible schedules when the second job is transferred to E

The objective function values of the schedules are as follows:

Z_{ik} = 10.1720

Z_{im} = 8.8542

Z_{ij} = 10.5139 *

Max $\{Z_{ik}, Z_{im}, Z_{ij}\} = Z_{ij}$. Transfer job j to early set. Then we have

d=2+6=8 and Z_2=10.5139.

Step 4. We again repeat steps 2 and 3 by transferring the third job to E. Schedules are given in Figure 3.7 after transferring the third job to E.

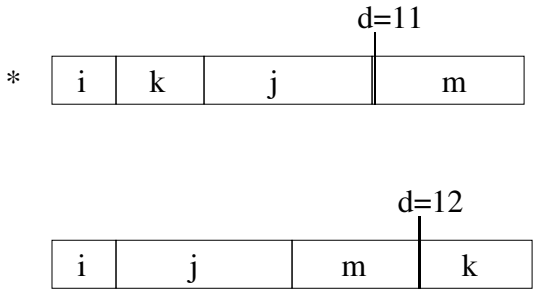

Figure 3.7. Transferring the jobs to E for finding the third job.

Objective function values:

Z_{ijk} = 10.9269 *

Z_{ijm}=9.4998

Max $\{Z_{ijk}, Z_{ijm}\}=Z_{ijk.}$ Transfer job k to E. Then, d=8+3=11 and Z_3=10.9269

Step 4. Only job m remains in the tardy set now. We repeat steps 2 and 3 by transferring it to E. Sequence the jobs as in Figure 3.8.

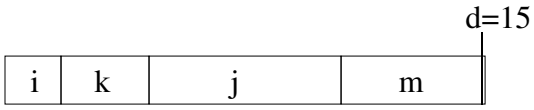

Figure 3.8 Ordering when all jobs in E

Objective function value Z_4 = 10.5152 and d = 11+4 = 15.

Step 5. $Z=max\{ Z_0, Z_1, Z_2, Z_3, Z_4\}=Z_3$. Hence, the best solution is the ordered early and tardy sets as E={i,k,j}, T={m} with due date *d*=11 and the best objective function value Z is 10.9269.

3.3.2 Upper Bound

We find an upper bound to the general version of the ETDR problem by relaxing the set definition by defining a single weight problem. The maximum of the e_i and t_i values is taken as the weight of job i. As a result, our problem reduces to the total discounted completion time problem with identical early and tardy weights, i.e. maximization of $\sum w_i \alpha^{C_i}$. The optimal solution to this problem can be found by sequencing the jobs in nonincreasing order of $\frac{w_i \alpha^{p_i}}{1-\alpha^{p_i}}$ ratios (see Theorem 2). Objective function of the upper bound problem is $Max \sum w_i \alpha^{C_i}$ where $w_i = \max\{e_i, t_i\}$.We formally state the procedure below:

Procedure UB

Step 1. Let w_i be $\max\{e_i,t_i\}$ for each job i.

Step 2. Sequence the jobs according to nonincreasing order of $R_{wi} = \frac{w_i \alpha^{p_i}}{1-\alpha^{p_i}}$ ratios.

Step 3. Objective function value of this sequence gives the upper bound.

Theorem 5 : Procedure UB gives a valid upper bound to the ETDR problem.

Proof: Suppose an optimal schedule S_1 in which job i is in the tardy set, and assume $\max\{e_i, t_i\}=e_i$. Let Z_{S_1} be the optimal objective function value for schedule S_1. Change t_i to e_i without changing the position of job i. Let the objective function value be Z_{S_2} for the new schedule. Note that $Z_{S_1} - Z_{S_2} = t_i \alpha^{C_i} - e_i \alpha^{C_i} < 0$. By repeating this process for all jobs, we can further increase the objective function value. This implies that Z_{S_2} provides an upper bound on Z_{S_1} value. After all changes, the solution is further improved by using the result of Theorem 1 without violating the validity of the upper bond.

□

We give the complexity result of UB procedure in Corollary 4.

Corollary 5 : Procedure UB runs in O(nlogn) time.

Proof: Complexity of UB is defined by forming $\frac{w_i \alpha^{p_i}}{1-\alpha^{p_i}}$ sequence in O(nlogn) steps.

□

3.3.3 Branch and Bound Algorithm

Each level of the branch and bound tree represents an assignment of a specific job to either set. As can be seen from Figure 3.9, for each job there are two decisions: either the job is in the early set, or it is in the tardy set. These decisions are taken according to the direction of our move. We define two moves in the tree: forward and backward. In a forward move, we assign a job to the early set E, while in a backward move we change the assignment of the job from the early set to the tardy set.

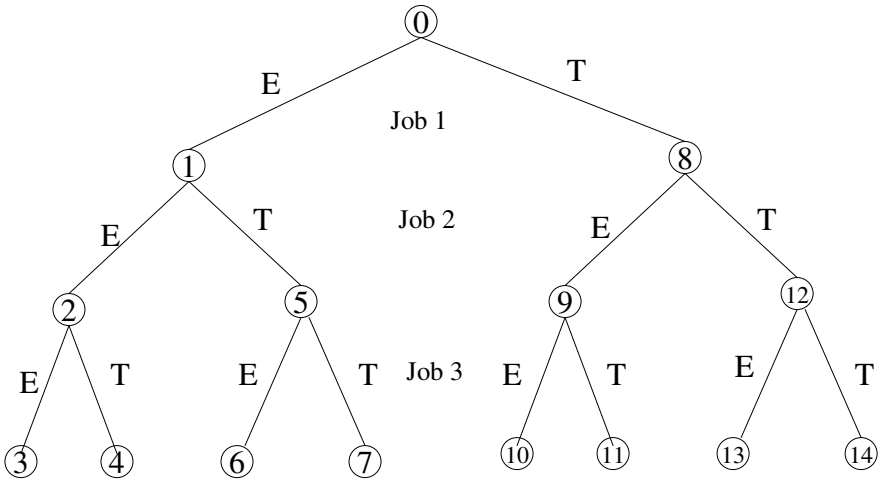

Figure 3.9 Branch and Bound Tree

Our branch and bound algorithm employs the depth-first strategy. According to this strategy, we go down by dealing with only one job at each level. We reach a complete schedule at the bottom of the tree. Figure 3.11 illustrates an instance with 3 jobs. After first 3 nodes are opened we have a schedule with all jobs in the early set. Then we backtrack. Nodes 1, 5 and 7 refer to a solution in which first job is in the early set, the others are free for node 1; the second job is tardy and the last job is free for node 5; and all jobs are fixed as job 1 is early, jobs 2 and 3 are tardy in node 7.

Lower bound is computed by using the Algorithm PR. Once a complete schedule is reached, the lower bound is updated if it is greater than the current best lower bound.

At each node, we calculate an upper bound as follows. We record the status of the jobs. If the job is in the early set, the weight w_i is recorded as e_i, if it belongs to tardy set, w_i is set to t_i. For the jobs whose sets are not determined yet, w_i is set to $\max\{e_i, t_i\}$ and these jobs are treated as if they are assigned to tardy set. Sequence of the jobs in each set is formed by Theorem 1.

Below is the stepwise description of our branching scheme:

Step 1. Calculate a lower bound value by using the Algorithm PR.

Step 2. Order the jobs according to nonincreasing R_{ei} values. Start from the first job in the order and assign it to the early set.

Step 3. Calculate the upper bound. If the upper bound for that node is greater than the current best lower bound, branch the node for the next job. Otherwise, fathom that node and backtrack to the first available job whose set is *E* and change its set status to *T*. Then repeat the process for this node.

Step 4. When last job of the sequence is reached, i.e. the sets of all jobs are determined, update the lower bound if the upper bound computed at this node is greater than the current best lower bound.
Step 5. Backtrack to the first available job whose set status is *E* and change its set to *T*. Repeat the process until no further backtracking is possible. (i.e. no job with set status E exists). The current lower bound is the optimal solution.

We now solve our example problem given in Section 3.3.1 by the B&B method.

Step 1. We have just found the lower bound as 10.9269 by using the Algorithm PR.

Step 2. R_{ei} ratios of the jobs are given in Table 3.1. The order of the jobs according to these ratios are as i,k,j,m. In node 1, assign job i to E. (Figure 3.10)

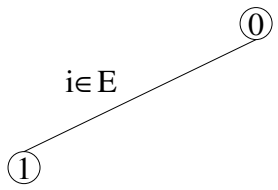

Figure 3.10 Branching node 1

Step 3. $E=\{i\}$, $T=\emptyset$, jobs k,m,j are free, i.e. their sets are not determined yet. We set $w_i=e_i$, $w_k=max\{e_k,t_k\}=7$, $w_j=max\{e_j,t_j\}=8$ and $w_m=max\{e_m,t_m\}=4$. The corresponding upper bound is $Z = 5\alpha^2 + 7\alpha^5 + 8\alpha^{11} + 4\alpha^{15} = 11.5174$ which is greater than our initial lower bound. So we go forward and set job k to E (Figure 3.11).

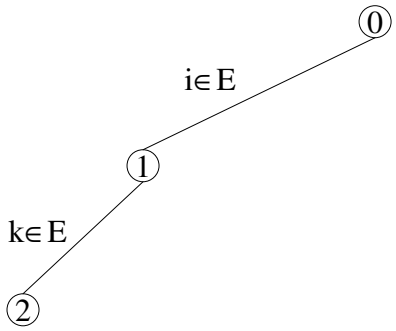

Figure 3.11. Branching node 2

In node 2, $E=\{i,k\}$, $T=\emptyset$, and the sets of jobs m and j are not determined yet. We set $w_i=e_i$, $w_k=e_k$, $w_j=max\{e_j,t_j\}$ and $w_m=max\{e_m,t_m\}$ and compute an upper bound as $Z=5\alpha^2+6\alpha^5+8\alpha^{11}+4\alpha^{15}=10.9269$ which is equal to our lower bound. We fathom node 2 and backtrack to the first available job and set job k to the tardy set. (Figure 3.12)

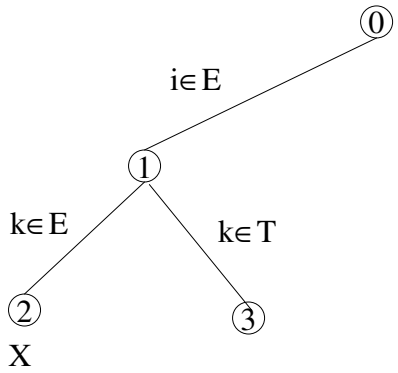

Figure 3.12. Backtracking to node 3

In node 3, $E=\{i\}$, $T=\{k\}$, jobs j and m are free. We set $w_i=e_i$, $w_k=t_k$, $w_j=max\{e_j,t_i\}$ $w_m=max\{e_m,t_m\}$. The upper bound is $Z=5\alpha^2+7\alpha^5+8\alpha^{11}+4\alpha^{15}=11.5174$ which is greater than the current lower bound. So go forward and assign job j to E. (Figure 3.13)

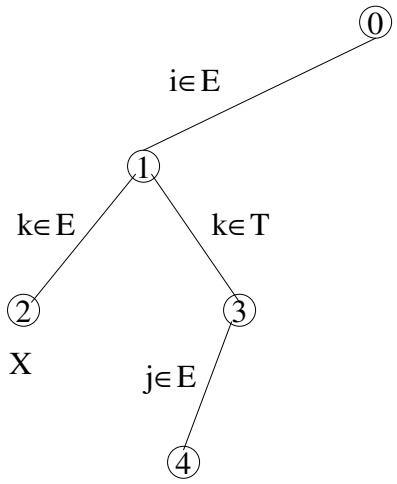

Figure 3.13 Branching node 4

In node 4, $E=\{i,j\}$, $T=\{k\}$, job m is free. We set $w_i=e_i$, $w_k=t_k$, $w_j=e_j$, $w_m=max\{e_m,t_m\}$. The upper bound is $Z=5\alpha^2+8\alpha^8+7\alpha^{11}+4\alpha^{15}=10.5139$ which is less than our best lower bound. So fathom node 4. Backtrack to the first available job, and set job j to tardy set.

In node 5, $E=\{i\}$, $T=\{j,k\}$, job m is free. We set $w_i=e_i$, $w_k=t_k$, $w_j=t_j$, $w_m=max\{e_m,t_m\}$ then compute the upper bound as $Z=5\alpha^2+7\alpha^5+4\alpha^9+5\alpha^{15}=10.7625$ which is less than the current lower bound. Fathom node 5 and backtrack to job i, then assign it to tardy set. (Figure 3.14)

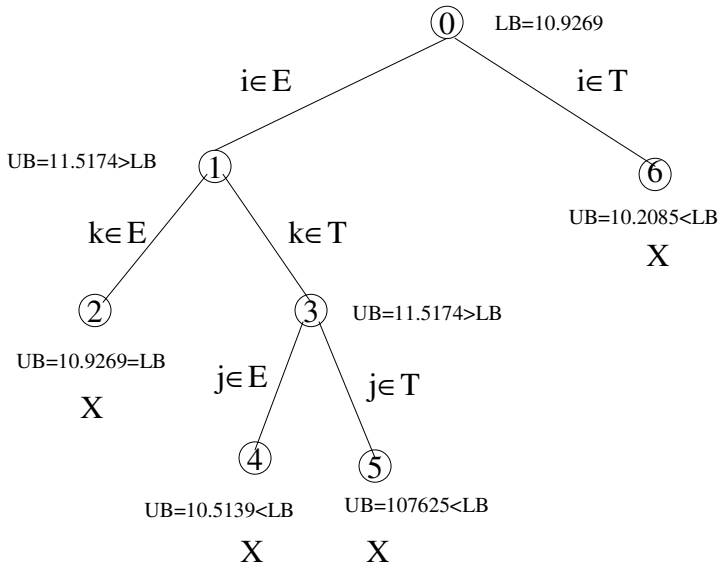

Figure 3.14 The complete tree

In node 6, $E= \emptyset$, $T=\{i\}$, the sets of jobs j, k, and m are not determined yet. We set $w_i=t_i$, $w_k=max\{e_k,t_k\}$ $w_j=max\{e_j,t_j\}$, $w_m=max\{e_m,t_m\}$, then compute the upper bound as $Z = 7\alpha^3 + 3\alpha^5 + 8\alpha^{11} + 4\alpha^{15} = 10.2085$ which is less than the current lower bound. Fathom node 6. There is no job for further backtracking. Our current lower bound is the optimal solution. We have computed this lower bound by using Algorithm PR. The schedule is i, k, j, m and i, k, and j are in the early set while job m is in the tardy set. Objective function value is 10.9269 and due date is that $d_i + d_k + d_j = 11$.

3.4 COMPUTATIONS

We code all algorithms in C language. A PC with a Pentium III 700 Mhz. processor is used to perform the computations.

We perform the computations with two groups of problem. Group 1 problems have p_i, e_i and t_i values that are uniformly distributed between 0 and 20, 0 and 15, 0 and 10, respectively. In practice, early incomes may be slightly higher than the tardy

incomes. One can think that finishing a job earlier is usually better, and generally a manufacturer charges a higher price for early completions. In this group, there are 6 sets of problems; each of them has 10 different instances. Totally we have 60 problems in Group 1.

We investigate the effect of processing time variability by generating Group 2 problems. Processing times are uniformly distributed between 0 and 100. We generate problem sets with 20, 50 and 100 jobs each having 10 different instances. Hence, we have 30 problems in Group 2.

Group 1 problems are taken from Biskup and Feldman (2001). Group 2 problems are generated by using a random number generator from internet web site Research Randomizer.

We find solutions to each problem instance for three different discount factor, α=0.9, 0.7 and 0.4 to see the effect of the discounting factor. We do not use α values that are less than 0.4 as small discount factors are not commonly observed in practice.

As a result we consider a total of 90 problems. For each problem, three different discount factors are considered, thereby resulting to a total of 270 problem instances.

Our performance measures are average and maximum percentage deviations from the upper bound; average and maximum number nodes in the B&B algorithm, average and maximum CPU times in seconds; and average and maximum due dates found. As our study is the first attempt to solve the problem, we do not have any benchmark study to compare our results. So we make comparison relative to upper bound values as shown below:

$$Deviation\ from\ UB(\%) = \frac{UB - OPT}{OPT} * 100$$

where OPT refers to the optimum solution found by B&B.

3.4.1. Results

The Algorithm PR and B&B have found exactly the same results for all problem instances. Average and maximum deviations from upper bounds are given in Table 3.2 for group 1 problems.

Table 3.2 Deviation from Upper Bound for Group 1 problems

Problem (N)	α=0.9		α=0.7		α=0.4	
	Average Deviation	Maximum Deviation	Average Deviation	Maximum Deviation	Average Deviation	Maximum Deviation
10	1.74	6.74	2.76	27.01	0.09	0.1
20	3.27	13.49	1.42	6.23	0.11	0.43
50	4.52	15.53	3.50	23.00	0.85	2.77
100	4.43	9.74	2.55	13.03	0.38	2.62
200	3.69	9.18	0.96	2.57	0.04	0.24
500	1.92	5.50	0.13	0.91	0	0

It can be seen from the table that as α decreases, both the average and maximum deviations from the upper bound decrease. When α values are relatively small, the positional changes of the jobs in the schedule cause small differences. Suppose a job with w_i=12, the completion times in schedule S_1 and S_2 are 18 and 34, respectively. When α=0.9 the difference in the objective function value of S_1 and S_2 is $12(\alpha^{18}-\alpha^{34})$= 1.4673 and when α=0.4, this difference reduces to 0.0000008. Deviations increase slightly as the number of jobs increases but when the number of jobs reaches to 100, it decreases again. This might be explained by the effect of discount factor. While the number of jobs increases, additive benefits of jobs positioned later become nearly zero, thereby causing less deviation.

Deviations of Group 2 problems are given in Table 3.3. Again both the average and maximum deviations decrease when α values get smaller. When compared with group 1 problems, deviations are smaller. Relatively greater p_i values lead to greater

completion times in Group 2 problems. Jobs at later positions of the schedule have nearly zero contribution to the objective value, so causing smaller deviations.

Table 3.3. Deviation from Upper Bound for Group 2 problems

N	α=0.9		α=0.7		α=0.4	
	Average Deviation	Maximum Deviation	Average Deviation	Maximum Deviation	Average Deviation	Maximum Deviation
20	1.33	9.85	0.33	3.17	0.04	0.04
50	4.16	16.65	1.40	3.05	0.05	0.13
100	3.85	15.68	3.41	14.27	2.27	13.50

Table 3.4 reports the average and maximum due dates for groups 1 and 2. Due dates are getting smaller while α values approaching to zero. Some jobs with high t_i values, tend to remain in the tardy set. But when discount factor approaches to zero they are likely to have a small completion time, decreasing due date forces them to remain in tardy set with this small completion time.

Table 3.4. Average and maximum due dates

Group	N	α=0.9		α=0.7		α=0.4	
		Av.	Max.	Av.	Max.	Av.	Max.
Group 1	10	24.5	63	12.3	34	7.5	13
	20	34.1	182	7.6	16	6.7	16
	50	29.2	56	6.7	15	4.5	11
	100	18.7	56	7.3	22	3	6
	200	28.8	83	12.7	36	5.4	15
	500	27	49	12	25	11.2	25
Group 2	10	29.9	84	20.8	84	16.8	49
	20	53.2	148	16.4	48	16.3	40
	50	46.9	119	18.7	54	16.2	38

Tables 3.5 and 3.6 give the average and maximum number of nodes and CPU times for Group 1 problems. Number of nodes and CPU times increases as the number of the jobs increases. Group 2 problems have the similar structure.

Table 3.5 Average and maximum number of nodes for Group 1 problems

N	α=0.9		α=0.7		α=0.4	
	Average # of nodes	Maximum # of nodes	Average # of nodes	Maximum # of nodes	Average # of nodes	Maximum # of nodes
10	21.8	28	24	40	25.8	84
20	44.8	54	40.2	42	86.26	98
50	150.4	264	102	110	109.4	194
100	271.6	378	221.8	390	219.6	390
200	1299.2	4238	465.1	832	445.4	794
500	5244.6	11180	1208.2	2002	1195.4	1978

Table 3.6 Average and maximum CPU times (seconds) for Group 1 problems

N	α=0.9		α=0.7		α=0.4	
	Average CPU Time	Maximum CPU Time	Average CPU Time	Maximum CPU Time	Average CPU Time	Maximum CPU Time
10	0.02	0.054	0.01	0.054	0.01	0.054
20	0.02	0.054	0.02	0.054	0.02	0.109
50	0.07	0.109	0.06	0.109	0.06	0.109
100	0.41	0.439	0.42	0.494	0.6	0.714
200	3.18	4.890	5.37	5.494	5.07	5.710
500	87.07	114.12	97.87	102.53	95	98.297

Our computational experiments reveal that the results of the Algorithm PR when processing times are arbitrary are the same with the B&B results for all problem instances. Our strong feeling is that the algorithm finds the optimal solution for the unrestricted due date ETDR problem when processing times are arbitrary. However, till now, we are unable to prove its validity and state the result as a conjecture. We present a methodology to prove the conjecture when p_i=1 in Appendix B.

Conjecture : Algorithm PR optimally solves the unrestricted due date ETDR problem.

CHAPTER 4

RESTRICTED COMMON DUE DATE

In this chapter, we consider the ETDR problem when the common due date is a parameter. In this case, we determine the contents of the early and tardy sets according to the given due date. The problem is to find the assignment of the jobs to sets E and T so as to maximize the total discounted revenue. We show that the problem is NP-hard through the reduction to a knapsack problem.

In the following section, three special cases are discussed. Due to the complexity of the problem, a branch and bound algorithm with lower and upper bound procedures is presented in Section 2. Computational results are given in the last section.

4.1. SPECIAL CASES

Three special cases which have been explained in Chapter 3 for the unrestricted due date problem are discussed for the restricted due date problem. We specify the first special case by setting the discount factor parameter to 1. Then we discuss a special case where early and tardy revenues are assumed to be constants. The third special case is found by setting the processing times to unity.

4.1.1. $\alpha=1$, p_i, e_i and t_i are arbitrary

In this case, we set the discount factor to 1 while keeping processing times, early and tardy revenues arbitrary. We show that our problem reduces to a knapsack problem with capacity *d*. The knapsack is defined by the early set. The benefit of putting item i into the knapsack is represented as $(e_i - t_i)$. Processing times are the capacity usages of the items that compete for the knapsack capacity. We aim to fill the knapsack, i.e. early set, so as to maximize total revenue. Below is the description of the model:

We let $X_i = \begin{cases} 1 & \text{if job i is assigned to early set} \\ 0 & \text{otherwise} \end{cases}$

The constraints are:

$\sum_{i=1}^{n} p_i X_i \le d$ (Knapsack capacity is not violated)

$X_i = 0$ or 1 (partial usages are not allowed)

The objective is:

Max Z, where $Z = \left[\sum_{i=1}^{n} e_i X_i + \sum_{i=1}^{n} t_i (1 - X_i) \right] = \left[\sum_{i=1}^{n} (e_i - t_i) X_i + \sum_{i=1}^{n} t_i \right]$

As $\sum_{i=1}^{n} t_i$ is constant, it can be taken out of maximization operator, and the objective can be simplified as:

$$Max \left[\sum_{i=1}^{n} (e_i - t_i) X_i \right]$$

Note that e_i-t_i represents the weight of job i, in the classical knapsack problem. Hence, a special case of the restricted due date ETDR problem can be formulated as an NP-hard 0-1 knapsack problem. It follows that the general ETDR problem is NP-hard as well.

4.1.2. $e_i=e$, $t_i=t$, p_i arbitrary and $0<\alpha<1$

In this special case, we assume all income values for early and tardy jobs are identical while keeping the processing times and the discount factor arbitrary. We define two cases: Case 1 $e \geq t$ and Case 2 $e<t$. We show that Case 1 is solved by putting the jobs to the early set by SPT rule as long as the common due date permits, then we continue putting the jobs to the tardy set by SPT rule. (see Theorem 7 below) However, if tardy income is greater than early income, i.e., Case 2, then the problem becomes NP-hard.

Theorem 6 : The Shortest Processing Time (SPT) rule finds the optimal solution to the restricted due date ETDR problem where $e_i=e$ and $t_i=t$ for all i and $e \geq t$.

Proof : From section 2.3.1, we know that nonincreasing order of $\frac{w_i \alpha^{p_i}}{1-\alpha^{p_i}}$ maximizes total revenue for a specified set of jobs. Note that, for any job i and k $\frac{\alpha^{p_i}}{1-\alpha^{p_i}} > \frac{\alpha^{p_k}}{1-\alpha^{p_k}}$ holds as $w_i=w_k$.

This follows that $\alpha^{p_i}(1-\alpha^{p_k}) > \alpha^{p_k}(1-\alpha^{p_i}) = \alpha^{p_i} - \alpha^{p_i+p_k} > \alpha^{p_k} - \alpha^{p_i+p_k}$

The last terms of the inequality cancel each other and we obtain $\alpha^{p_i} > \alpha^{p_k}$. Hence $\ln \alpha^{p_i} < \ln \alpha^{p_k}$, which implies $p_i < p_k$.

Suppose a schedule S_1 shown as Figure 4.1 is optimal where $p_i<p_k<p_m$. We will show that by reordering the jobs in S_1, we can get a better schedule which contradicts with the optimality of S_1.

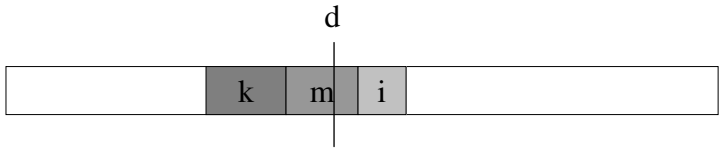

Figure 4.1. Ordering of jobs i,k and m in Schedule S_1

After ordering the jobs according to SPT, we get the schedules S_2^a and S_2^b as shown in Figure 4.2.

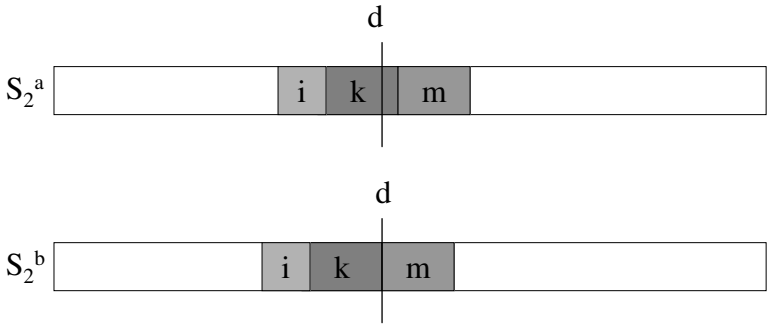

Figure 4.2. Ordering the jobs according to SPT

There may be two cases for job k: It may be transfered to the tardy set due to sliding back as in schedule S_2^a or it may remain in the early set as in schedule S_2^b. Completion times of the remaining jobs other than job i, k and m are not affected from this change. Let c(S_1), c(S_2^a) and c(S_2^b) be the contributions of job i, k and m to the objective functions of the schedules S_1, S_2^a and S_2^b respectively:

$$c(S_1) = e\alpha^{C+p_k} + t\alpha^{C+p_k+p_m} + t\alpha^{C+p_k+p_m+p_i}$$

$$c(S_2^a) = e\alpha^{C+p_i} + t\alpha^{C+p_i+p_k} + t\alpha^{C+p_k+p_m+p_i}$$

$$c(S_2^b) = e\alpha^{C+p_i} + e\alpha^{C+p_i+p_k} + t\alpha^{C+p_k+p_m+p_i}$$

where C is the completion time of the last job before job i.

Dividing all terms by α^C and canceling the third terms end up with the following terms:

$c(S_1) \rightarrow e\alpha^{p_k} + t\alpha^{p_k+p_m}$

$c(S_2^a) \rightarrow e\alpha^{p_i} + t\alpha^{p_i+p_k}$

$c(S_2^b) \rightarrow e\alpha^{p_i} + e\alpha^{p_i+p_k}$

Note that $p_i<p_k<p_m$. First, we compare $c(S_1)$ with $c(S_2^a)$. $e\alpha^{p_i} < e\alpha^{p_k}$ as $p_i<p_k$. $p_i+p_k<p_k+p_m$, implies that $t\alpha^{p_i+p_k} < t\alpha^{p_k+p_m}$. Hence, $c(S_1)<c(S_2^a)$. Now, we compare $c(S_1)$ with $c(S_2^b)$. $e\alpha^{p_i} < e\alpha^{p_k}$ as $p_i<p_k$. Then $p_i+p_k<p_k+p_m$ and $e>t$, imply that $t\alpha^{p_i+p_k} < e\alpha^{p_k+p_m}$, which follows $c(S_1)<c(S_2^b)$. As a result, $c(S_1)$ is less than $c(S_2^a)$ and $c(S_2^b)$ which contradicts with the optimality of schedule S_1.

□

We give the running time of the algorithm in Corollary 6.

Corollary 6: If $e_i=e$, $t_i=t$ for all i and $e \geq t$, then the problem is solved in O(nlogn) time.

Proof: The optimal solution to this problem can be found in O(nlogn) time, i.e. the time needed to sort the jobs by SPT rule.

□

4.1.3 $p_i = 1$, e_i and t_i are arbitrary, $0<\alpha<1$

When all processing times are identical, our problem reduces to a linear assignment problem as in Chapter 3. Note that the job assigned to position j will be completed at time j, and there will be a job completed exactly at due date d. We should decide to assign which job to which position so as to maximize total revenue. The difference from the unrestricted due date case is that assignment problem is solved only for the

specified due date. Mathematical representation of this linear assignment problem is given below:

Let $X_{ij} = \begin{cases} 1 & if \ \ job\ i\ assigned \ \ position \ \ j \\ 0 & otherwise \end{cases}$

The objective function of the assignment problem is

$$Max \sum_{ij} c_{ij} X_{ij}$$

where c_{ij} is the revenue of assigning job i to position j. Note that $c_{ij}=e_i\alpha^j$ if $i \in E$ and $c_{ij}=t_i\alpha^j$ if $i \in T$. Hence, the objective function reduces to:

$$Max\left[\sum_{\substack{ij \\ i \in E}} e_i \alpha^j X_{ij} + \sum_{\substack{ij \\ i \in T}} t_i \alpha^j X_{ij} \right]$$

The assignment constraints are:

$$\sum_{i=1}^{n} X_{ij} = 1 \qquad \text{j = 1,2.....,n}$$

$$\sum_{j=1}^{n} X_{ij} = 1 \qquad \text{i = 1,2.....,n}$$

$$X_{ij} = 0-1 \qquad \text{i = 1,2.....,n} \quad \text{j = 1,2.....,n}$$

Corollary 7 : The restricted due date ETDR problem is solved $O(n^3)$ steps when $p_i=1$.

Proof : The assignment algorithm can be solved in $O(n^3)$ time (see, for example Janker and Volgenant (1987)). The solution requires optimal solution of assignment algorithm for the specified d.

□

4.2. SOLUTION APPROACHES TO THE RESTRICTED DUE DATE ETDR PROBLEM

We show that when α=1, the problem reduces to the well-known NP-hard knapsack problem. Hence the general problem with arbitrary α value is NP-hard as well. In this section, we will present a heuristic algorithm and a branch and bound algorithm to solve this NP-hard problem. The algorithms are similar to the ones presented for unrestricted due date problem.

4.2.1 Lower Bound

We use the Algorithm PR to find a lower bound for the restricted due date ETDR problem. We assume if a job enters early set, it never leaves. The algorithm returns the solution when due date is reached. Below is the stepwise description of the procedure:

Step 1. Set $d'=0$. Set $T=\{i/i\in N\}$ and $E=\emptyset$. Find $R_{ti} = \frac{t_i \alpha^{p_i}}{1-\alpha^{p_i}}$ ratio for each job i and order the jobs according to nonincreasing R_{ti} values.

Step 2. Calculate the objective function value when each job is transferred one by one from T to E. Determine the job that gives the maximum objective value and assign it to E. Let this job be r. Update d' as $d'=d'+p_r$. Set $E=E\cup\{r\}$ and $T=T\backslash\{r\}$

Step 3. If $d' \geq d$ then compute the objective function value and stop, else go to step 2.

4.2.2 Upper Bound

An upper bound to the general version of ETDR problem is found by relaxing the set definition and by defining a single weight problem. The maximum of the e_i and t_i values is taken as the weight of the job i. As a result, our problem reduces to the total discounted completion time problem with identical early and tardy weights, i.e.

maximization of $\sum w_i \alpha^{C_i}$ where $w_i = \max\{e_i, t_i\}$. The optimal solution to this problem is formed by sequencing the jobs in nonincreasing order of $\frac{w_i \alpha^{p_i}}{1-\alpha^{p_i}}$ ratios. (see Theorem 2) Objective function of the upper bound problem is $Max \sum w_i \alpha^{C_i}$ where $w_i = \max\{e_i, t_i\}$.We formally state the procedure below:

One may refer to Chapter 3 for the proof and the complexity result of UB procedure.

4.2.3 Branch and Bound Algorithm

Depth first technique is used in our branch and bound algorithm as in Chapter 3. Each level i represents the assignment of job i and each job i has two alternatives: Alternative *E* means that job i is in early set, and alternative *T* means, that job i is in tardy set. Similar to the solution of a knapsack problem, we try to fill the early set as much as we can so as to maximize total revenue. The difference from the knapsack problem is the existence of the tardy set. In knapsack problems, items that are not covered by the knapsack do not contribute to the objective function. However in our problem we should decide on the sequencing of jobs in the tardy set, as well.

Lower bound is calculated by using Algorithm PR. At each node we calculate an upper bound by recording the status of the jobs. If the job is in the early set, the weight w_i is recorded as e_i, if it belongs to the tardy set, then w_i is set to t_i. For the jobs whose sets are not determined yet, w_i is set to $\max\{e_i, t_i\}$. We assume that these jobs are assigned to the tardy set and sequenced according to Theorem 1.

Processing times of the jobs that are in the early set are summed at each node. When we reach the specified due date, the objective function value is calculated by assuming the remaining jobs are in tardy set. If the objective function value is greater than the current best lower bound, we update the best lower bound value. We then backtrack to the first available job that is assigned to set *E*. Stepwise description of the procedure is given below:

Step1. Calculate the lower bound by Algorithm PR.

Step 2. Order the jobs according to nonincreasing value of R_{ie} values. Start from the first job and assign it to the early set. Let $d_t = \sum_{i \in E} p_i$. If $d_t < d$ then calculate the upper bound. If upper bound of a node is greater than the current best lower bound branch for the next job. Otherwise fathom that node and backtrack to the first available job which has status *E* and change the status by opening a *T* node for that job. Repeat the process for this node.

Step 3. If $d_t = d$ assign the remaining jobs to set *T*. Calculate *Z* value. If *Z* value is greater than the current best lower bound, update the current best lower bound. Go to step 5.

Step 4. If $d_t > d$, assign the current job and the remaining jobs to tardy set and compute the *Z* value. If the *Z* value is greater than the current best lower bound, update the lower bound. Go to Step 5.

Step 5. Backtrack to the first available job having status *E* and change its status to *T*. Repeat the process until you cannot backtrack further. The current lower bound is the optimal solution.

Now we demonstrate the lower bound and B&B algorithms on the 4-jobs problem instance given in Chapter 3. We assume α=0.9 again and d=$0.2\sum_{i \in N} p_i$ = 0.2*15=3.

Table 4.1 reports the data for this example. R_{ei} and R_{ti} ratios are computed as $\frac{e_i \alpha^{p_i}}{1-\alpha^{p_i}}$ and $\frac{t_i \alpha^{p_i}}{1-\alpha^{p_i}}$, respectively and tabulated on the table.

Table 4.1. Data for the example problem

job	e_i	t_i	p_i	R_{ei}	R_{ti}
i	5	3	2	21.31	12.78
j	8	5	6	9.07	5.67
k	6	7	3	16.14	18.83
m	2	4	4	3.81	7.63

Step 1. We find the lower bound by using the heuristic algorithm. Let d'=0 and $R_{ti} = \frac{t\alpha^{p_i}}{1-\alpha^{p_i}}$ values are computed for each job and given in Table 4.1. The sequence of the jobs is given in Figure 4.3

k	i	m	j

Figure 4.3. Sequence of jobs when d'=0

The objective value of the sequence is $Z_0=t_k\alpha^{Ck}+ t_i\alpha^{Ci}+ t_m\alpha^{Cm}+ t_j\alpha^{Cj}$ =9.4536

Transfer each job one by one to the early set as shown in Figure 4.4, and come up with four schedules generated temporarily. Note that when jobs m and j are assigned to early set, d' becomes greater than the specified due date. So jobs m and j are treated as they are in the tardy set and we get the same schedule when d'=0. We do not compute objective function value of these schedules.

$Z_k = e_k\alpha^{Ck}+ t_i\alpha^{Ci}+ t_m\alpha^{Cm}+ t_j\alpha^{Cj}=$ 8.7246

$Z_i = e_i\alpha^{Ci}+ t_k\alpha^{Ck}+ t_m\alpha^{Cm}+ t_j\alpha^{Cj}=$ 10.7625 *

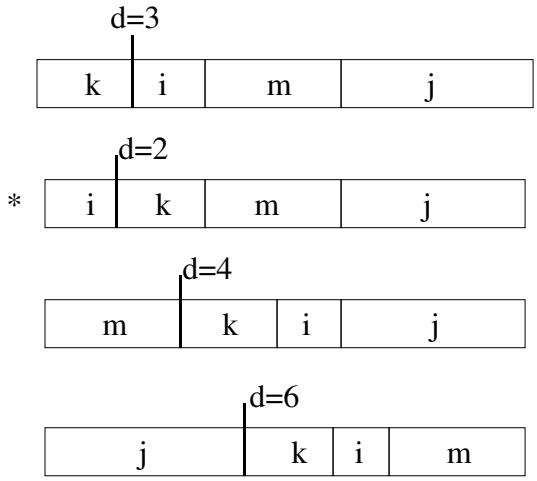

Figure 4.4. Transferring jobs *T* to *E*

Since $Max\ \{Z_k, Z_i\} = Z_i$ we transfer job i to the early set.

Set $E=\{i\}$ and $T=\{k,m,j\}$, and set d=2 and Z_1=10.7625

Repeat Steps 2 and 3 of the heuristic algorithm for transferring the second job to *E*. Three schedules to be evaluated are given in Figure 4.5. When we assign the second job to the early set, *d'* becomes greater than *d*. So the result that found in the previous step returns the solution of the heuristic algorithm as below:

Set $E=\{i\}$ and $T=\{k,m,j\}$, the objective function value is *Z*=10.7625

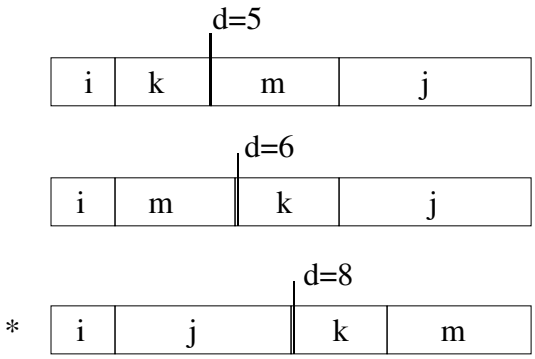

Figure 4.5. Three possible schedules when the second job is transferred to *E*

Step 2. R_e ratios of the jobs are given in Table 4.1. The order of the jobs according to these ratios are i,k,j,m. In node 1, set job i to *E*. The complete tree is given in Figure 4.6.

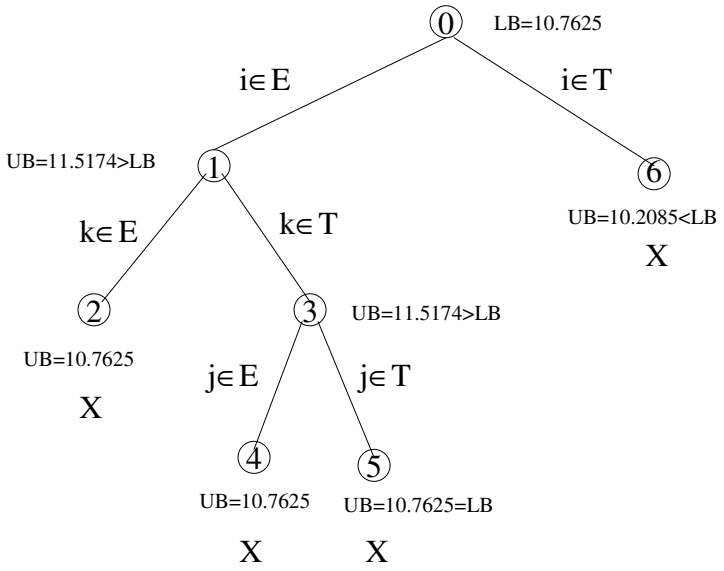

Figure 4.6 The complete branch and bound tree for the example problem

Now, $d_t=p_i=2<d=3$. Then, $E=\{i\}$, $T=\emptyset$, and sets of jobs k,m,j are not determined yet. So we set $w_i=e_i$, $w_k=max\{e_k,t_k\}$, $w_j=max\{e_j,t_j\}$ and $w_m=max\{e_m,t_m\}$. The corresponding upper bound is $Z=5\alpha^2+7\alpha^5+8\alpha^{11}+4\alpha^{15}=11.5174$ which is greater than our initial lower bound. So we go forward and assign job k to E.

Step 4. In node 2 $d_t=p_i+p_k=5>d$, then assign the current job k and the remaining jobs j and m to the tardy set. Set $E=\{i\}$, $T=\{k,j,m\}$. We set $w_i=e_i$, $w_k=t_k$, $w_j=t_i$ $w_m=t_m$. The upper bound is $Z=5\alpha^2+7\alpha^5+4\alpha^9+5\alpha^{15}=10.7625$ which is equal to current lower bound.

Step 5. In node 3, backtrack to job k. $d_t=p_i=2<$d. Set $E=\{i\}$, $T=\{k\}$, jobs j and m are free. We set $w_i=e_i$, $w_k=t_k$, $w_j=max\{e_j,t_i\}$, and $w_m=max\{e_m,t_m\}$. The upper bound is $Z=5\alpha^2+7\alpha^5+8\alpha^{11}+4\alpha^{15}=11.5174$ which is greater than current lower bound. So go forward and assign job j to E.

In node 4, $d_t=p_i+p_j=8>d$. Assign the current job j and job m to the tardy set. Set $E=\{i\}$, $T=\{k,j,m\}$. We set $w_i=e_i$, $w_k=t_k$, $w_j=t_i$, and $w_m=t_m$. The upper bound is $Z=5\alpha^2+7\alpha^5+4\alpha^9+5\alpha^{15}=10.7625$ which is equal to the current lower bound.

Step 5. Backtrack to job j. In node 5, $d_t=p_i=2<d$. Set $E=\{i\}$, $T=\{j,k\}$, job m is free. We set $w_i=e_i$, $w_k=t_k$, $w_j=t_j$, $w_m=max\{e_m,t_m\}$ then compute the upper bound as $Z=5\alpha^2+7\alpha^5+4\alpha^9+5\alpha^{15}=10.7625$, which is equal to current lower bound. Fathom node 5 and backtrack to job i, and assign it to the tardy set.

In node 6, $d_t=0<d$. Set $E=\emptyset$, $T=\{i\}$, and sets of jobs j, k, and m are not determined yet. We set $w_i=t_i$, $w_k=max\{e_k,t_k\}$ $w_j=max\{e_jt_j\}$, $w_m=max\{e_m,t_m\}$ then compute the upper bound as $Z=7\alpha^3+3\alpha^5+8\alpha^{11}+4\alpha^{15}=10.2085$ which is less than current lower bound. Fathom node 6. There is no available job for backtracking. Our current lower bound, by chance, returns the optimal solution.

4.3. COMPUTATIONS

We code all algorithms in C language. A PC with a Pentium III 700 Mhz. processor is used to perform computations.

We perform the computations with three groups of problem. Group 1 problems have p_i, e_i and t_i values that are uniformly distributed between 0 and 20, 0 and 15, 0 and 10, respectively. In this group, there are 6 sets of problems; each of them has 10 different instances. Totally we have 60 problems in Group 1.

We investigate the effects of processing time variability by generating Group 2 problems. We generate processing times uniform between 0 and 100. We generate problem sets with 20 and 50 jobs and each having 10 different instances. Hence we have 20 problems in Group 2.

Group 3 problems are generated to see the changes in the results when e_i and t_i values are correlated. For this purpose, we use $t_i = 0.75\ e_i$ and set n=20 and 50 jobs and for each n 10 instances are generated again. There are totally 20 problems in Group 3. Remember that we did not perform computations with group 3 problems for unrestricted due date case. We know that when e_i and t_i values are correlated, all the jobs are assigned to early or tardy set in unrestricted case.

Group 1 problems have been taken from Biskup and Feldman (2001). Group 2 and 3 problems are generated by using a random number generator from internet web site Research Randomizer.

We set d to three different values as $d=0.8\sum p_i$, $d=0.5\sum p_i$ and $d=0.2\sum p_i$. By using 0.8 and 0.2 coefficients, we aim to see the effects of dominant tardy and early sets. 0.5 represents a balanced set definition. For each due date setting, computations are performed for three different discount rates as α=0.9, 0.7 and 0.4.

As a result we consider a total of 100 problems. For each problem instance, we perform nine computations formed by three different discount factors and three due date coefficients, resulting 900 computations.

We take the deviation of the lower bound by using the formula:

$$Deviation\ \ from\ OPT(\%) = \frac{OPT - LB}{OPT} * 100$$

where OPT refers to the optimal solution by B&B.

The deviations of the upper bounds are also computed by using the formula:

$$Deviation\ From\ OPT(\%) = \frac{UB - OPT}{OPT} * 100$$

The deviations are considered both in average and maximum terms. Finally, we find B&B performance by recording the average and maximum number of nodes and CPU times. As our study is the first attempt in the literature we do not have any benchmark study for comparison.

4.3.1 Results

Lower bounds found by Algorithm PR deviate from the optimal solutions by branch and bound algorithm only for 16 problem instances. Table 4.2 gives these specific problems. As can be seen from the table, even these deviations are negligibly small. All cases occur when discount factor is 0.9. As explained in Chapter 3, when discount factor decreases, the contribution of the jobs to the objective function value that are ordered at later positions of the schedule approaches to zero. Also when we use 0.7 and 0.4 as discount factors, the effects of positional difference become smaller. In 13 out of 16 cases, due date coefficient is 0.2. 13 of these deviations belong to group 1 problems. 3 of them belong to group 3 problems, i.e. correlated e and t values. No deviation exists among group 2 problems.

Table 4.2 Cases in which Lower Bounds deviate from Branch & Bound results when $\alpha = 0.9$

Problem Instances	Due Date Coefficient	Z Value		Deviation From OPT (%)
		LB	B&B	
10-8	0.8	20.887234	20.895220	0.03
20-4	0.5	16.529919	16.529948	0.0001
20-7	0.5	13.739237	13.739240	0.00002
10-10	0.2	15.390556	15.454228	0.41
20-1	0.2	22.869563	22.968755	0.43
20-6	0.2	30.162419	30.172029	0.03
20-9	0.2	36.527710	36.534364	0.018
50-1	0.2	42.020299	42.020321	0.00005
50-2	0.2	32.526508	32.526551	0.0001
50-5	0.2	34.977059	34.977102	0.0001
50-6	0.2	37.265692	37.265740	0.0001
50-7	0.2	29.973329	29.973363	0.0001
50-10	0.2	37.956328	37.956338	0.00002
20-9	0.2	36.374503	36.373651	0.00002
50-5	0.2	34.976977	34.976979	0.000005
50-10	0.2	37.956188	37.956210	0.000003

Average and maximum deviations of the upper bounds for group 1 problems are presented for due date coefficients of 0.2, 0.5 and 0.8 in Table 4.3.

Table 4.3 Deviations from the Upper Bound for Group 1 problems

Due Date	N	α=0.9		α=0.7		α=0.4	
		Average Deviation	Maximum Deviation	Average Deviation	Maximum Deviation	Average Deviation	Maximum Deviation
0.2	10	6.00	26.96	7.29	47.31	47.50	471.0
	20	5.54	24.37	15.31	120.1	45.27	405.1
	50	5.39	16.18	7.80	51.71	11.74	106.9
	100	9.68	29.78	11.80	45.34	16.67	83.57
	200	5.11	9.96	2.94	4.48	0.55	2.79
	500	2.53	6.80	0.58	3.77	0.04	0.39
0.5	10	6.03	28.09	7.29	47.31	47.50	471.0
	20	5.62	24.35	15.31	120.1	45.27	405.1
	50	5.39	16.18	7.80	51.71	11.74	106.9
	100	9.68	29.78	11.80	45.34	16.67	83.57
	200	5.11	9.96	2.94	4.48	0.55	2.79
	500	2.53	6.80	0.58	3.77	0.04	0.39
0.8	10	6.03	28.09	7.29	47.31	47.50	471.0
	20	5.62	24.35	15.31	120.1	45.27	405.1
	50	5.39	16.18	7.80	51.71	11.74	106.9
	100	9.68	29.78	11.80	45.34	16.67	83.57
	200	5.11	9.96	2.94	4.48	0.55	2.79
	500	2.53	6.80	0.58	3.77	0.04	0.39

When we compare the results with that of unrestricted case, both the average and maximum upper bound deviations increase. In restricted case, the number of nodes by B&B algorithm is also higher. When the number of jobs increases, CPU time required by the B&B algorithm increases. Hence we could solve the problems having 10, 20 and 50 jobs with Branch and Bound algorithm. Problems with 100, 200 and 500 jobs are solved by using Algorithm PR, therefore the deviations for the problem sets of 100, 200 and 500 are computed by using the heuristic results.

As can be seen from the tables, contrary to the unrestricted due date case, the deviations increase with a decrease in the discount factor. Exceptions are due to the problems having 200 and 500 jobs. These problem instances have relatively smaller deviations. The reason can be explained by the exponential nature of the objective function. The completion times of the jobs at later of the schedule are so large that their contribution to the objective value approaches to zero.

Examining the tables with different due date coefficients reveals that the deviations are not affected by due date coefficients. Slight changes may be observed in problem sets of 10 and 20 jobs and with 0.9 and 0.7 discount factors. Discount factors dispel the effect of the due dates significantly.

Deviations from the upper bound for Group 2 and 3 problems are presented for each due date coefficient in Table 4.4.

Table 4.4 Deviations from the Upper Bound for Groups 2 and 3

Due Date	Group	N	α=0.9		α=0.7		α=0.4	
			Average Deviation	Maximum Deviation	Average Deviation	Maximum Deviation	Average Deviation	Maximum Deviation
0.2	2	20	3.21	19.81	2.06	16.65	0.52	2.91
		50	12.63	49.82	22.48	49.82	50.84	197.5
	3	20	0.11	0.40	0	0	0	0
		50	0.0001	0.0002	0	0	0	0
0.5	2	20	3.21	19.81	2.06	16.65	0.52	2.91
		50	12.63	49.82	22.48	49.82	50.84	197.5
	3	20	0.0004	0.003	0	0	0	0
		50	0	0	0	0	0	0
0.8	2	20	3.21	19.81	2.06	16.65	0.52	2.91
		50	12.63	49.82	22.48	49.82	50.84	197.5
	3	20	0	0	0	0	0	0
		50	0	0	0	0	0	0

Group 2 problems have higher deviations compared to the unrestricted case. The deviations are not affected by the due date coefficients due to the discount factor. The deviations tend to decrease when discount factor gets smaller. Exception is due to the problems having 50 jobs. However, there are two problems having deviations greater than 50% at discount rate 0.4, thereby raising the average deviations.

Most of the deviations of Group 3 problems are zero. When due date coefficients are 0.2 and discount factor is 0.9, very slight deviations are observed. We know that Group 3 problems have correlated e and t values, so upper bound and unrestricted case optimal solution coincide with all jobs in the early set. But even if $d = 0.2\sum_i p_i$, we have slight deviations for α=0.9 and no deviations for α=0.7 and 0.4. As we have explained before, discount factor diminishes the effect of the due dates.

The average and maximum number of nodes and the CPU times are presented according to the due date coefficients in Tables 4.5 and 4.6.

Table 4.5 Average and maximum number of nodes

Due date	Group	N	α=0.9		α=0.7		α=0.4	
			Average # of nodes	Maximum # of nodes	Average # of nodes	Maximum # of nodes	Average # of nodes	Maximum # of nodes
0.2	1	10	28	42	35.1	61	33.4	47
		20	199.6	399	147.5	246	139.1	187
		50	7769.7	32477	3797.1	15178	3710629	11503303
	2	20	205.7	420	851.7	2002	5606	8614
		50	302532	1897786	697136	1215554	8815414	11773977
	3	20	40.5	59	35	39	39.4	62
		50	100.8	131	1963	15138	8249	58351
0.5	1	10	66.4	139	49	97	60.2	98
		20	455.2	1154	246.8	372	4255	7373
		50	23427	52724	223152	358126	7243559	16334325
	2	20	577.5	1287	25647	75928	113573	171648
		50	2715172	8526938	19235036	38328948	98415676	167891406
	3	20	40.8	45	70.8	165	5046.6	16120
		50	121.1	136	91318	810942	1795262	7188655
0.8	1	10	87.6	178	60	101	90.2	174
		20	596.7	1928	597.9	1085	9654.3	16250
		50	430660	3373233	3780125	9127883	28246658	54726653
	2	20	1395	4020	43907	130175	175880	260808
		50	8369301	25087476	33921709	67041114	157540080	268268280
	3	20	40	47	419.1	859	9551	16126
		50	396251	3551373	484636	4132391	18152662	132003961

Table 4.6 Average and Maximum CPU times (Seconds)

Due date	Group	N	α=0.9		α=0.7		α=0.4	
			Average CPU Time	Maximum CPU Time	Average CPU Time	Maximum CPU Time	Average CPU Time	Maximum CPU Time
0.2	1	10	0	0	0	0	0	0.054
		20	0.0054	0.054	0.0054	0.054	0.005	0.054
		50	0.456	1.813	0.236	0.879	212.9	809.4
	2	20	0.01	0.054	0.032	0.054	0.142	0.274
		50	15.55	97.58	33.51	64.83	1080	6571
	3	20	0.02	0.02	0.01	0.05	0.02	0.05
		50	0.04	0.05	0.109	0.824	0.241	1.648
0.5	1	10	0.054	0.054	0.054	0.054	0.0054	0.054
		20	0.01	0.054	0.01	0.054	0.0604	0.109
		50	1.142	2.582	19.45	121.1	178.8	554
	2	20	0.02	0.054	0.367	1.483	2.554	4.065
		50	68.15	213.07	619.4	1318.85	3792.3	6571
	3	20	0	0	0.02	0.05	0.065	0.109
		50	0.054	0.054	1.966	17.41	35.835	147.80
0.8	1	10	0.109	0.054	0.0054	0.054	0.0054	0.054
		20	0.021	0.054	0.016	0.054	0.1043	0.219
		50	11.64	87.03	290.5	1216	1052.2	2736
	2	20	0.05	0.054	0.582	1.978	3.247	5.109
		50	163.23	494.45	930.8	1993.08	5106.1	8874.7
	3	20	0.02	0.05	0.03	0.05	0.109	0.21
		50	10.71	89.06	7.81	66.04	492.2	4123

When compared with the unrestricted due date case both the number of nodes and CPU time values are larger. Larger deviations between upper bound and optimal solution cause an increase in the number of nodes, thereby increasing the CPU times. Group 2 problems have the worst performance compared to Group 1 and 3 due to very large completion times caused by high variability of processing times.

CHAPTER 5

CONCLUSIONS

In this study, we develop a scheduling model based on monetary issues. Maximizing the net present value of the revenues earned at the completion of each job, is the objective function of the model. There are two revenue alternatives for each job, depending on the completion time. If a job is completed before or exactly at the due date, an early income is received; otherwise a tardy income is received.

We have studied the problem in single machine environments. We assume a common due date for all jobs. Common due date is the most important parameter leading two different problem classes. When the due date is treated as a decision variable, it is referred to as unrestricted due date. If the due date is specified by the customer then we call it restricted due date. We have studied these two cases separately.

We show that the complexity of the unrestricted due date ETDR problem is open, and develop polynomial time algorithms for several special cases. These special cases are those with constant early and tardy revenues, unit processing times, and discount factor that equals to 1. Due to the complexity of the problem, we develop a B&B algorithm and enhance its efficiency with lower and upper bounds.

Computational experiments reveal that all the results for 270 problem instances by B&B and our lower bound are the same. This lead us to present a conjecture that states the optimality of lower bound for unrestricted due date ETDR problem. We observe from our experiments that as discount factor getting smaller, the upper bound deviations decrease. Decreasing discount factor pushes the due date forward as the jobs having high tardy incomes are likely to have small completion times and remain in tardy set. Highly variable processing times cause smaller deviations.

For restricted due date ETDR problem, we study the special cases as well. We show that our problem reduces to knapsack problem when $\alpha=1$. In case of constant early and tardy incomes, SPT finds the optimal solution if early income is greater. On the other hand, we give an optimal solution procedure when the processing times are unity. We modify our B&B algorithm for unrestricted due date case to restricted due date, and use the same upper and lower bounds.

Our computational results show that Algorithm PR deviates from the optimal solution only for 16 out of all problem instances. These deviations are even less than 1%. We also observe that restricting the due date increases the upper deviations when discount factor decreases on contrary to the unrestricted due date case. Another important result is that the discount factor diminishes the effect of due date settings. Even if in case of the early and tardy incomes are correlated, restricting the due date does not change the results and no deviation occurs.

Finally, we can say that Algorithm PR is effective for solving both versions of the general problem. We are able to solve the problems having up to hundreds of jobs by using this algorithm in reasonable time.

To the best of our knowledge, our study is a first attempt to solve a single machine maximal revenue problem with discounted early and tardy incomes. There are a plenty of future research areas that are worth-studying. Some these are listed below:

- Distinct due date values for the jobs

- Parallel machine environments
- Stochastic processing times

REFERENCES

1. Aggarwal S.C., Wyman F.D. (1973) An investigation of a cost based rule for job shop scheduling. International Journal of Production Research, 11(3) : 247-261.

2. Aggarwal S.C., Mc Cart B.A. (1974) The development and evaluation of a cost based composite scheduling rule. Naval Research Logistics Quarterly, 21(1): 155-169.

3. Amar A.D., Xiao B. (1997) Scheduling on a bottleneck station: A comprehensive cost model and heuristic algorithm. International Journal of Production Research, 35(4): 1011-1030.

4. Baker K.R., Scudder G.D. (1990) Sequencing with earliness and tardiness penalties: A review. Operations Research, 38(1): 22-34.

5. Benton W.C., (1993) Time and cost based priorities for job shop scheduling. International Journal of Production Research, 31(7): 1509-1519.

6. Biskup D., Feldman M. (2001) Benchmarks for scheduling on a single machine against restrictive and unrestrictive common due dates. Computers and Operations Research, 28: 787-801.

7. Blank L.T., Tarquin A.J. (1998) Engineering Economy, McGraw-Hill Publications.

8. Doersch R.H., Patterson J.H. (1977) Scheduling a project to maximize its present value. Management Science, 23(8): 882-889.

9. Hall N.G., Posner M.E. (1991) Earliness-tardiness scheduling problems I: Weighted deviation of completion times about a common due date. Operations Reseach, 39(5): 836-846.

10. Hall N.G., Kubiak W., Sethi S.P. (1991) Earliness-tardiness scheduling problems II: Deviation of completion times about a restrictive common due date. Operations Reseach, 39(5): 847-856.

11. Heerolen W.S., Dommelon P.V. (1997) Project network models with discounted cash flows. A guided tour through recent developments. European Journal of Operational Research, Vol 100: 97-121.

12. Hoffman T.R., Scudder G.D. (1983) Priority scheduling with cost considerations. International Journal of Production Research, 21(6): 881-889.

13. Janker R., Volgenant A. (1987) A shortest augmenting path algorithm for dense and sparse linear assignment problems. Computing, 8: 325-340.

14. Jones J.J. (1971) Optimization of work-in-process inventories in job shops. Division Research Harvard University Graduate School of Business Administration HBS 71-4.

15. Jones C.H. (1973) An economic evaluation of job shop dispatching rules. Management Science, 20(3): 293-307.

16. Wilson H.G., Mardis D.S. (1983) Modifying job sequencing rules for work-in-process inventory reduction. IIE Transactions, 15(4): 320-323.

17. Pinedo M. (1995) Scheduling : Theory, algorithms, and systems. Prentice Hall.

18. Russel A.H. (1970) Cash flows in networks. Management Science, 16(5): 357-373.

19. Yang K.K., Talbot F.B., Patterson J.H. (1992) Scheduling a project to maximize its net present value: an integer programming approach. European Journal of Operational Research, 64: 188-198.

APPENDIX A

Table A. 1 Special cases for restricted due date ETDR

p_i	e_i	t_i	α	Solution
1	arb*	arb	arb	Assignment Algorithm
1	arb	t	arb	Nonincreasing order of e_i. After filling set E, assign others to set T arbitrarily.
1	e	arb	arb	Nondecreasing order of t_i. After filling set E starting from the first job, assign the others according to nonincreasing order of t_i values to set T.
1	e	t	arb	Trivial
Arb	arb	arb	1	Knapsack problem
Arb	arb	t	1	Knapsack problem
Arb	e	arb	1	Knapsack problem
Arb	e	t	1	If e>t SPT If e<t Longest Processing Time (LPT)
1	arb	arb	1	add according to nonincreasing order of (e_i-t_i) values to E
1	arb	t	1	add according to nonincreasing order of (e_i-t) values to E
1	e	arb	1	add according to nonincreasing order of $(e-t_i)$ values to E
1	e	t	1	Trivial
arb	arb	arb	arb	NP-hard
arb	arb	t	arb	NP-hard
arb	e	arb	arb	NP-hard
arb	e	t	arb	If e>t SPT . If e<t NP-hard

Table A.2 Special cases for unrestricted due date ETDR

p_i	e_i	t_i	α	Solution
1	arb	arb	arb	Assignment Algorithm
1	arb	t	arb	Assignment Algorithm
1	e	arb	arb	Assignment Algorithm
1	e	t	arb	Algorithm SA
arb	arb	arb	1	Algorithm BW
arb	arb	t	1	Algorithm BW
arb	e	arb	1	Algorithm BW
arb	e	t	1	Algorithm SA
1	arb	arb	1	Algorithm BW
1	arb	t	1	Algorithm BW
1	e	arb	1	Algorithm BW
1	e	t	1	Algorithm SA
arb	arb	arb	arb	Open
arb	arb	t	arb	Open
arb	e	arb	arb	Open
arb	e	t	arb	Algorithm SA

* arb: arbitrary

APPENDIX B

We give a conjecture at the end of Chapter 3 which states that Algorithm PR finds the optimal solution of the general ETDR problem. Here, we present a methodology that could be used to prove this conjecture when $p_i=1$. The proof remains incomplete, but it may be helpful for a future proof attempt.

Conjecture : Algorithm PR optimally solves the unrestricted due date ETDR problem.

Proof : Assume that early set is not empty, i.e. $j \in E$ for a subset of jobs. Form three schedules S^i, S^k and S^m where jobs i, k and m are in the early set, respectively. Let Z^i be the total revenue obtained by schedule S^i; Z^k by schedule S^k and Z^m by schedule S^m. Suppose that S^i is the optimal schedule and Z^i is greater than Z^k and Z^m. By transferring one more job from tardy set to the early set, we form new schedules S^{ik}, S^{im} and S^{km}. S^{ik} is formed by transferring job k to the early set while preserving other jobs in E; schedule S^{im} by transferring job m to the early set and again preserving other jobs in E; schedule S^{km} by transferring job k and m to the early set and job i to the tardy set. Let Z^{ik} be the total revenue obtained by schedule S^{ik}, Z^{im} by schedule S^{im} and Z^{km} by schedule S^{km}. To prove the theorem, it should be shown that $\max\{Z^{ik}, Z^{im}\}$ is greater than Z^{km}.

We have 3 different cases due to the positions of the jobs in the early set. In case 1, the position of job i is between job k and m. The position of job i is later than job k and m in case 2, and before than these two jobs in case 3. For each these 3 cases, we have 6 sub-cases according to the ordering of the jobs in the tardy set.

Case 1

Case 1 represents the ordering of jobs i, k and m in the early set as given in Figures B.1 and B.2

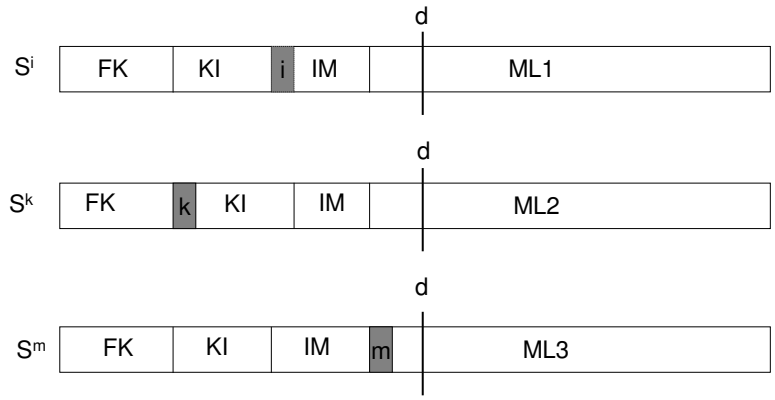

Figure B.1 Ordering of jobs in early set for schedules S^i, S^k and S^m.

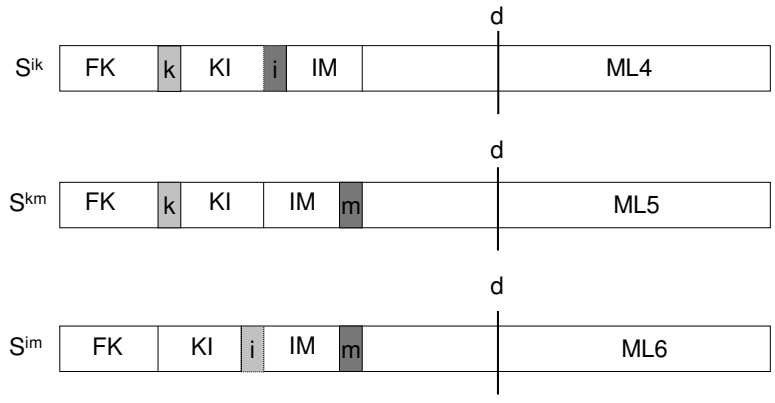

Figure B.2 Ordering of the jobs for schedules S^{ki}, S^{km} and S^{im}.

As can be seen from the figures, we partition the schedule into subsets by excluding jobs i, k and m. FK represents the set of jobs starting from the first job until job k, KI is the set of jobs between job k and i, and IM is the set of jobs between i and m. ML represents the set of jobs from m through the last job. Note that the job contents of the part ML is different for each of the schedule. For example, in schedule S^i, ML1 contains job k and m in the tardy set while in schedule S^k, ML2 contains job i and m in the tardy set.

We use the following representation to compute the objective function values. We denote the total revenues provided by the jobs in set FK as $w_{FK}\alpha^{C_{FK}}$ which equals to $\sum_{i \in FK} w_i \alpha^{C_i}$. We write the objective functions for schedules S^i, S^k and S^m by using this representation as:

$$Z^i = w_{FK}\alpha^{C_{FK}} + w_{KI}\alpha^{C_{KI}} + e_i\alpha^{C_i} + w_{IM}\alpha^{C_{IM}+1} + w_{ML1}\alpha^{C_{ML1}+1}$$

$$Z^k = w_{FK}\alpha^{C_{FK}} + e_k\alpha^{C_k} + w_{KI}\alpha^{C_{KI}+1} + w_{IM}\alpha^{C_{IM}+1} + w_{ML2}\alpha^{C_{ML2}+1}$$

$$Z^m = w_{FK}\alpha^{C_{FK}} + w_{KI}\alpha^{C_{KI}} + w_{IM}\alpha^{C_{IM}} + e_m\alpha^{C_m} + w_{ML3}\alpha^{C_{ML3}+1}$$

From the optimality of schedule S^i, we know that the difference between the objective values of schedule S^i and schedule S^k, i.e., $\Delta_1 = Z^i - Z^k > 0$, or between the objective values of schedule S^i and schedule S^m is positive. i.e. $\Delta_1 = Z^i - Z^m > 0$

Objective function values for schedules S^{ik}, S^{km} and S^{im} are given below:

$$Z^{ik} = w_{FK}\alpha^{C_{FK}} + e_k\alpha^{C_k} + w_{KI}\alpha^{C_{KI}+1} + e_i\alpha^{C_i+1} + w_{IM}\alpha^{C_{IM}+2} + w_{ML4}\alpha^{C_{ML4}+2}$$

$$Z^{km} = w_{FK}\alpha^{C_{FK}} + e_k\alpha^{C_k} + w_{KI}\alpha^{C_{KI}+1} + w_{IM}\alpha^{C_{IM}+1} + e_m\alpha^{C_m+1} + w_{ML5}\alpha^{C_{ML5}+2}$$

$$Z^{im} = w_{FK}\alpha^{C_{FK}} + w_{KI}\alpha^{C_{KI}} + e_i\alpha^{C_i} + w_{IM}\alpha^{C_{IM}+1} + e_m\alpha^{C_m+1} + w_{ML6}\alpha^{C_{ML6}+2}$$

Let Δ_2 represent the difference between objective value, Z^{im}, of schedule S^{im} and objective value, Z^{km}, of schedule S^{km}, i.e., $\Delta_2 = Z^{im} - Z^{km}$, or objective value, Z^{ik}, of schedule S^{ik} and objective value, Z^{km}, of schedule S^{km} i.e. $\Delta_2 = Z^{ik} - Z^{km}$.

If we can show that Δ_2 is positive by comparing Δ_1 for each of the sub-cases, we will complete the proof for case 1.

Case 2

Case 2 represents the ordering of the jobs i, k and m in the early set as given in Figures B.3 and B.4

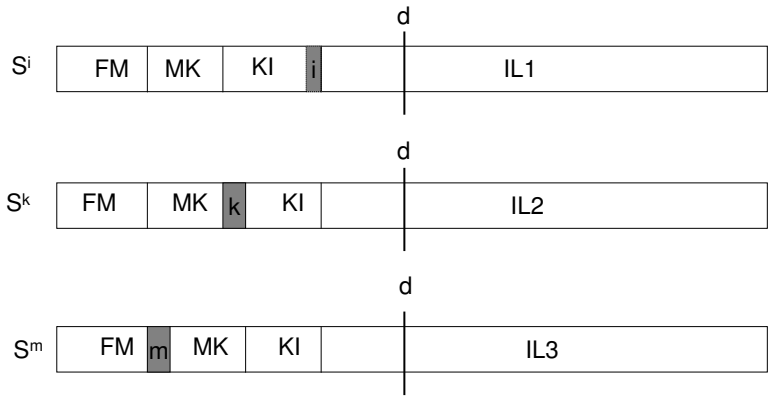

Figure B.3 Ordering of jobs in early set for schedules S^i, S^k and S^m.

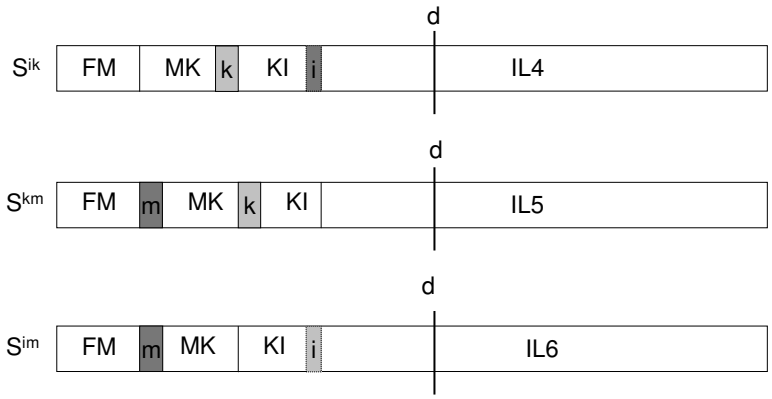

Figure B.4 Ordering of the jobs for schedules S^{ki}, S^{km} and S^{im}.

As can be seen from the figures, we partition the schedule into subsets by excluding jobs i, k and m. FM represents the set of jobs from the first one until job m; MK the set of jobs between job m and k and KI is the set of jobs between k and i. IL represents the set of jobs from k through the last job. Note that the job contents of the part IL is different for each of the schedule. For example, in schedule S^i, IL1 contains job k and m in the tardy set while in schedule S^k, IL2 contains job i and m in the tardy set.

We write the objective functions for schedules S^i, S^k and S^m as follows:

$$Z^i = w_{FM}\alpha^{C_{FM}} + w_{MK}\alpha^{C_{MK}} + w_{KI}\alpha^{C_{KI}} + e_i\alpha^{C_i} + w_{IL1}\alpha^{C_{IL1}+1}$$

$$Z^k = w_{FM}\alpha^{C_{FM}} + w_{MK}\alpha^{C_{MK}} + e_k\alpha^{C_k} + w_{KI}\alpha^{C_{KI}+1} + w_{IL2}\alpha^{C_{IL2}+1}$$

$$Z^m = w_{FM}\alpha^{C_{FM}} + e_m\alpha^{C_m} + w_{MK}\alpha^{C_{MK}+1} + w_{KI}\alpha^{C_{KI}+1} + w_{IL3}\alpha^{C_{IL3}+1}$$

From the optimality of schedule S^i, we know that the difference between the objective values of schedule S^i and schedule S^k, i.e., $\Delta_1 = Z^i - Z^k > 0$, or between the objective values of schedule S^i and schedule S^m is positive. i.e. $\Delta_1 = Z^i - Z^m > 0$

Objective functions for schedules S^{ik}, S^{km} and S^{im} are given below:

$$Z^{ik} = w_{FM}\alpha^{C_{FM}} + w_{MK}\alpha^{C_{MK}} + e_k\alpha^{C_k} + w_{KI}\alpha^{C_{KI}+1} + e_i\alpha^{C_i+1} + w_{IL4}\alpha^{C_{IL4}+2}$$

$$Z^{km} = w_{FM}\alpha^{C_{FM}} + e_m\alpha^{C_m} + w_{MK}\alpha^{C_{MK}+1} + e_k\alpha^{C_k+1} + w_{KI}\alpha^{C_{KI}+2} + w_{IL5}\alpha^{C_{IL5}+2}$$

$$Z^{im} = w_{FM}\alpha^{C_{FM}} + e_m\alpha^{C_m} + w_{MK}\alpha^{C_{MK}+1} + w_{KI}\alpha^{C_{KI}+1} + e_i\alpha^{C_i+1} + w_{IL6}\alpha^{C_{IL6}+2}$$

Let Δ_2 represent the difference between objective value, Z^{im}, of schedule S^{im} and objective value, Z^{km}, of schedule S^{km}, i.e., $\Delta_2 = Z^{im} - Z^{km}$, or objective value, Z^{ik}, of schedule S^{ik} and objective value, Z^{km}, of schedule S^{km} i.e. $\Delta_2 = Z^{ik} - Z^{km}$.

If we can show that Δ_2 is positive by comparing Δ_1 for each of the sub-cases, we will complete the proof for case 2.

Case 3

Case 3 represents the situation that the ordering of the jobs i, k and m in the early set is as in Figures B.5 and B.6

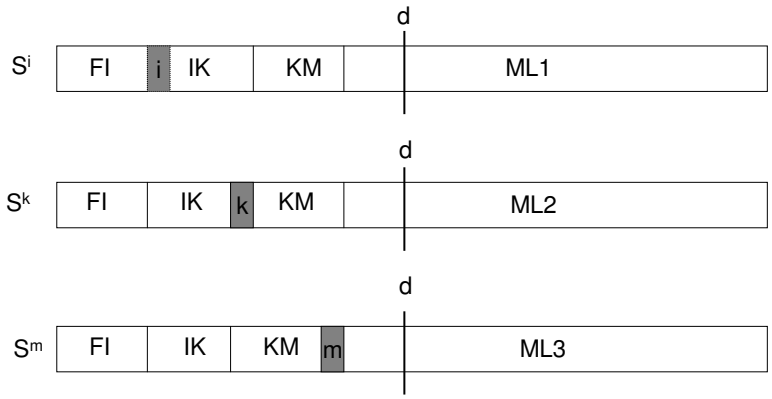

Figure B.5 Ordering of jobs in early set for schedules S^i, S^k and S^m.

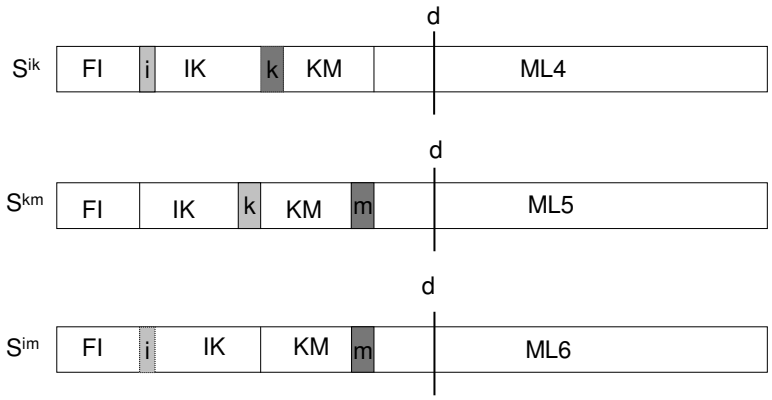

Figure B.6 Ordering of the jobs for schedules S^{ki}, S^{km} and S^{im}.

As can be seen from the figures, we partition the schedule into subsets by excluding jobs i, k and m. FI represents the set of jobs from the first one until job i; IK the set of jobs between job i and k and KM is the set of jobs between k and m. ML represents the set of jobs from m through the last job. Note that the job contents of the part ML is different for each of the schedule. For example, in schedule S^{ik}, ML4 contains job m in the tardy set while in schedule S^{km}, ML2 contains job i in the tardy set.

We write the objective functions for schedules S^i, S^k and S^m as below:

$$Z^i = w_{FI}\alpha^{C_{FI}} + e_i\alpha^{C_i} + w_{IK}\alpha^{C_{IK}+1} + w_{KM}\alpha^{C_{KM}+1} + w_{ML1}\alpha^{C_{ML1}+1}$$

$$Z^k = w_{FI}\alpha^{C_{FI}} + w_{IK}\alpha^{C_{IK}} + e_k\alpha^{C_k} + w_{KM}\alpha^{C_{KM}+1} + w_{ML2}\alpha^{C_{ML2}+1}$$

$$Z^m = w_{FI}\alpha^{C_{FI}} + w_{IK}\alpha^{C_{IK}} + w_{KM}\alpha^{C_{KM}} + e_m\alpha^{C_m} + w_{ML3}\alpha^{C_{ML3}+1}$$

From the optimality of schedule S^i, we know that the difference between the objective values of schedule S^i and schedule S^k, i.e., $\Delta_1 = Z^i - Z^k > 0$, or between the objective values of schedule S^i and schedule S^m is positive. i.e. $\Delta_1 = Z^i - Z^m > 0$

Objective functions for schedules S^{ik}, S^{km} and S^{im} are given below:

$$Z^{ik} = w_{FI}\alpha^{C_{FI}} + e_i\alpha^{C_i} + w_{IK}\alpha^{C_{IK}+1} + e_k\alpha^{C_k+1} + w_{KM}\alpha^{C_{KM}+2} + w_{ML4}\alpha^{C_{ML4}+2}$$

$$Z^{km} = w_{FI}\alpha^{C_{FI}} + w_{IK}\alpha^{C_{IK}} + e_k\alpha^{C_k} + w_{KM}\alpha^{C_{KM}+1} + e_m\alpha^{C_m+1} + w_{ML5}\alpha^{C_{ML5}+2}$$

$$Z^{im} = w_{FI}\alpha^{C_{FI}} + e_i\alpha^{C_i} + w_{IK}\alpha^{C_{IK}+1} + w_{KM}\alpha^{C_{KM}+1} + e_m\alpha^{C_m+1} + w_{ML6}\alpha^{C_{ML6}+2}$$

Let Δ_2 represent the difference between objective value, Z^{im}, of schedule S^{im} and objective value, Z^{km}, of schedule S^{km}, i.e., $\Delta_2 = Z^{im} - Z^{km}$, or objective value, Z^{ik}, of schedule S^{ik} and objective value, Z^{km}, of schedule S^{km} i.e. $\Delta_2 = Z^{ik} - Z^{km}$.

If we can show that Δ_2 is positive by comparing Δ_1 for each of the sub-cases, we will complete the proof for case 3.

□