A PROJECT PAYMENT SCHEDULING PROBLEM WITH DISCOUNTED CASH FLOWS

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

BY

ALİCAN CÖMERT

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING

JULY 2014

Approval of the thesis:

A PROJECT PAYMENT SCHEDULING PROBLEM WITH DISCOUNTED CASH FLOWS

Submitted by **ALİCAN CÖMERT** in partial fulfillment of the requirements for the degree of **Master of Science in Industrial Engineering Department, Middle East Technical University** by,

Prof.Dr. Canan Özgen _____
Dean of Graduate School of **Natural and Applied Sciences**

Prof. Dr. Murat Köksalan _____
Head of Department, **Industrial Engineering**

Prof. Dr. Meral Azizoğlu _____
Supervisor, **Industrial Engineering Department, METU**

Examining Committee Members:

Assoc. Prof. Dr. Canan Sepil _____
Industrial Engineering Department, METU

Prof. Dr. Meral Azizoğlu _____
Industrial Engineering Department, METU

Assist. Prof. Dr. Özgen Karaer _____
Industrial Engineering Department, METU

Assoc. Prof. Dr. Ferda Can Çetinkaya _____
Industrial Engineering Department, Çankaya Üniversitesi

Hasan Haluk Kobakçı, M.Sc. _____
Director, ASELSAN A.Ş.

Date: 08.07.2014

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Alican CÖMERT

Signature :

ABSTRACT

A PROJECT PAYMENT SCHEDULING PROBLEM WITH DISCOUNTED CASH FLOWS

Cömert, Alican

M.Sc., Department of Industrial Engineering
Supervisor: Prof. Dr. Meral Azizoğlu

July 2014, 72 pages

In this study we consider a project payment model with discounted cash flows. We assume that the client payment times are defined in the project contract. The activities are characterized by their processing times and costs that are incurred at their completions. Our problem is to find the client payment amounts and activity completion times so as to minimize the net present value of the client payments and activity costs. We show that the problem is strongly NP-hard.

We formulate the problem as a mixed integer nonlinear programming model and solve small to moderate sized problem instances. For moderate to large sized problem instances, we propose a branch and bound algorithm that employs efficient lower and upper bounding mechanisms.

Keywords: project scheduling, discounted cash flows, branch and bound algoritm

ÖZ

İSKONTOLANDIRILMIŞ NAKİT AKIŞLI PROJE ÖDEME ÇİZELGELEMESİ PROBLEMİ

Cömert, Alican
Yüksek Lisans, Endüstri Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Meral Azizoğlu

Temmuz 2014, 72 sayfa

Bu çalışmada, nakit akışlarının iskontolandırıldığı bir proje çizelgeleme problemi ele alınmıştır. Müşteri ödemelerinin tanımlı zamanlarda ve proje harcamalarının ise aktivite bitiş noktalarında yapıldığı varsayılmıştır. Problemimiz, toplam müşteri ödeme ve aktivite maliyetlerinin bugünkü değerini ençoklayan, müşteri ödeme miktarlarını ve aktivite bitiş sürelerini belirlemektir. Problemimizin NP-zor olduğunu gösterdik.

Problemimizi tam sayılı karmaşık doğrusal olmayan bir model marifetiyle tanımaya çalıştık. Modelin küçük boyutlu problemler için optimal çözümü bulduğunu gördük. Orta boyutlu problemleri çözebilmek için bir dal-sınır algoritmasını geliştirdik. Algoritmanın performansı optimal çözümün özelliklerini ve geliştirdiğimiz alt ve üst sınırlama mekanizmalarını kullanarak iyileştirdik

.

Anahtar Kelimeler: proje çizelgelendirmesi, iskontolandırılmış nakit akışları, dal-sınır algoritması

To my family

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor Prof. Dr. Meral Azizoğlu for her invaluable guidance, encouragements, advices and insight during the entire study. Without her hard work, understanding and support, it would be impossible to complete this thesis.

I would also like to thank the respected committee members, for accepting to participate in committee and their valuable comments and suggestions.

I would like to give my appreciation to my company, ASELSAN A.Ş. and my managers for supporting me and giving the permission to attend such a great MSc. program.

Also, thanks to my friends Eyüp Ensar Altaş, Kerem Erkan, Şemsettin Balta, Onur Ata, Uğur Koç and Umutcan Duman for their true friendship and sharing joyful times with me throughout my thesis period.

Sincere thanks to my family, especially my mother Alise Cömert, my father İsmet Cömert, and my sister Aylin Bekiroğlu for their love, support and patience to me. I could not have finished this study without them by my side.

Finally, I must express my great love and appreciation to my dearest fiancée Naz Güler, for her love, patience, support and being with me whenever I need.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Project is a temporary group activity designed to produce a unique product, service or result (Project Management Institute). Project management is the application of knowledge skills and techniques to execute the project effectively and efficiently. Financial planning plays a vital role in efficient and effective project management by providing powerful means of using the cash flows, by considering the time value of money.

The basic concern of the financial planning is to increase the profitability of the projects via establishing payment schedules that are acceptable by the contractors (producers) and the clients (customers). The payment schedules should consider the timing and quantity of each receipt from the client and the timing and amount of each expense by the contractor. The expenses made by the contractor usually due to the activity realizations. The objective of the financial planners is usually to maximize the net present worth of all cash flows. From the contractor side, the negative cash flow is due to the activity expenses and the positive cash flow is the amount received from the client to realize the project.

During the bid submission process, the contractor and client should negotiate over the payment schedule proposals. The client would like to pay the money as late as possible and the contractor would like to receive the money as early as possible. A good payment schedule would consider the compromises between the client and contractor, requirements.

In the literature, there are mainly four payment models that are lump-sum payment, payments at event occurrences, equal time intervals and progress payment. Lump sum payment model is rather easy to solve for the net present value problems compared to other payment models, since the client pays the whole amount to the contractor at the end of the project. Thus, the objective function reduces to finishing the project as early as possible. In the payments at event occurrences model, payments are made at predefined events where timing and amount of payments are to be ascertained. For equal time intervals model, payments are made in equal intervals whereas the last payment is done with the completion of the project. Progress payment model is very similar to the equal time intervals model with a significant difference that the payment times are not equally spaced and number of payments that will be made during the project is unknown.

In this study, we use progress payment scheduling model in which the timings of the payments are already specified but their amounts are going to be determined. We assume that the activity costs are incurred at activity completion times. In practice, the majority of the activities are outsourced or subcontracting, hence the cost is charged once the activity is received the complete form. Our objective is to maximize the net present worth of all cash flows of the contractor. From the contractor side, the negative cash flow is due to activity expenses and the positive cash flow is the amount received from the client for the work completed after the last payment.

Our model and solution approaches can be used by the client side, after some modifications. We hope our study could help for the practitioners who want to manage their cash flows, negotiating for the milestones and critical control points, determining the amounts and timings of the progress payments.

In this study, we present a branch and bound algorithm that decides on the activity completion times and the amount of progress payments. We assume in the contract, the deadline of the project completion time is already negotiated. To the best of our knowledge, our study is the first attempt to solve the problem to optimality.

The rest of the thesis is organized as follows. Section 2 defines the problem and presents the mathematical model. The complexity of the problem is shown as NP-hard in the strong sense. Some feasible solutions and an optimal solution of the model are also illustrated. Section 3 gives the survey of the related literature. In Section 4, we discuss our branch and bound algorithm together with the bounding mechanisms. The results of our extensive computational runs are reported in Section 5. Section 6 concludes the study by pointing out our main findings and suggestions for future research.

CHAPTER 2

PROBLEM DEFINITION

In this chapter, with its underlying assumptions together, a mixed integer non-linear mathematical model is presented. Then, the complexity of the problem is settled. Finally, a feasible solution and an optimal solution of the model are discussed.

2.1 Problem Statement

The problem is analyzed from the contractor's side. The objective of the problem is to maximize the net present value of all cash flows which consist of contractor's cost of activities and client's payments.

We consider a progress payment model where the project payments occur at predefined dates. The amount of money to be paid is calculated according to the expenses of the completed activities in that period. Considering the time value of money, the client wishes to pay as late as possible, whereas the contractor wishes to receive payments as early as possible, in the meantime there are activity expenses of contractor and a project deadline which contractor need to finish all activities before that date.

We assume the client and contractor has an agreed schedule for the payment dates. On the predefined dates, the client pays the activity costs that are finished between the last payment and payment date, with a predefined profit margin over the activity costs. The profit margin is bargained at the beginning of the project

and it might change from period to period. For example, if the client is willing to receive the project as soon as possible, the contractor may be awarded by setting higher profit margins to earlier periods.

Contractor's costs are assumed to be charged at the end of activity in a lump-sum. If an activity does not finish before a payment date, although it has started before the payment date, the contractor neither receives the payment of that activity nor pays its cost in that payment interval.

Discount factor is the maximum rate which the contractor could invest their money with. This rate may be the market interest rate or external rate of return of the contractor. Discount factor may change from period to period. Therefore, interest rate forecasts during the project timeline could be reflected to project the costs.

In the study, we assume that the interest rate and profit margin are static parameters. We further make the following assumptions:

- Project activities, activity durations, activity costs and their precedence relationships are deterministic, i.e., known with certainty.

- All predecessor relations have finish-to-start dependency which means an activity cannot start before its all preceding activities are not finished.

- Once an activity starts, it must be completed without any interruption.

- There is no resource constraint.

- The discount rate is continuously compounded.

- Profit margin includes cost of activities, therefore, should be greater than 1.

2.2 Mathematical Model

We assume that there are N activities and P payment periods and use the following notation.

Parameters:

i : Index for activities, $i = 1,\ldots,N$

p : Index for progress payments, $p = 1,\ldots,P$

A_p : Finishing time of period p

α_p : Discount factor for period p, we assume $\alpha_p = \alpha$ for $\forall\ p$

D : Deadline of the project

d_i : Duration of activity i

β_p : Profit margin for period p, we assume $\beta_p = \beta$ for $\forall\ p$

C_i : Cost of activity i (negative cash flow)

IP_i: Immediate predecessor of activity i

Decision variables:

CF_p : Amount of payment received at the end of period p, $p = 1,\ldots,P$ (positive cash flow)

T_i : Completion time of activity i, $i = 1,\ldots,N$

X_{pi}: $\begin{cases} 1 \text{ If activity } i \text{ ends within period } p \\ 0 \text{ otherwise} \end{cases}$

Note that we assume the timing of the payments and amount of the activity costs are given, i.e., parameters. However, the timing of the activity completions and amount of progress payments are to be decided, i.e., decision variables.

The cash flow diagram that explains our problem is given in Figure 1.

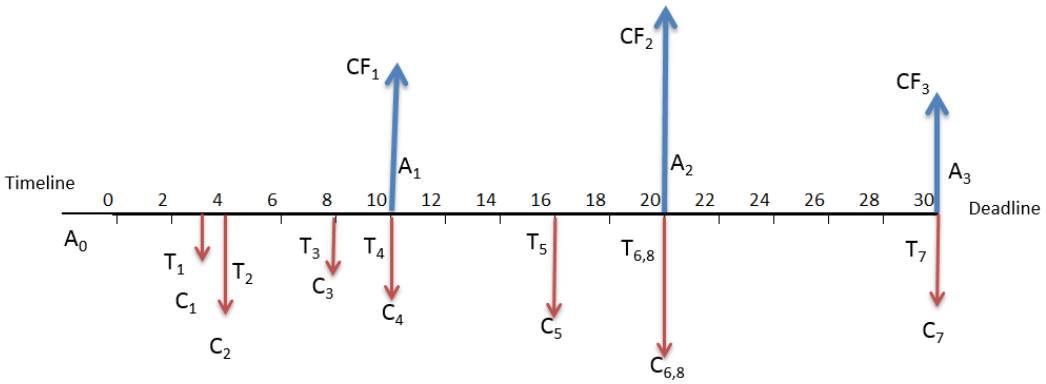

Figure 1 - The Cash Flow Diagram

Model:

Max $\sum_{p=1}^{P} exp(-\alpha.A_p).CF_p - \sum_{i=1}^{N} C_i.exp(-\alpha.T_i)$

subject to

(1) $T_i \geq T_j + d_i$ for $\forall\, j \in IP_i$, $\forall\, i = 1,\ldots,N$

(2) $T_{N+1} \leq D$

(3) $T_0 = 0$

(4) $\sum_{p=1}^{P} X_{pi} = 1$ for $\forall\, i$, $i = 1,\ldots,N$

(5) $T_i \leq \sum_{p=1}^{P} A_p.X_{pi}$ for $\forall\, i$, $i = 1,\ldots,N$

(6) $T_i \geq \sum_{p=1}^{P} A_{(p-1)}.X_{pi}$ for $\forall\, i$, $i = 1,\ldots,N$

(7) $\sum_{i=1}^{N} \beta . C_i . X_{pi} = CF_p$ for $\forall\, p$, $p = 1,\ldots,P$

(8) $X_{pi} = 0$ or 1 for $\forall\, i, p$

The model above is a mixed integer non-linear problem. It is written from the point of view of the contractor. The objective function has two separate parts: The money that contractor receives from the client considering the time value of money and the expenses of activities of the contractor considering the time value of money. Note that the second part is a function of T_is.

We now explain our constraints.

(1) Activity predecessor constraints guarantee that an activity cannot start before all its preceding activities are finished.
(2) The project must be finished before the specified deadline.
(3) Activity 0 is the first activity that defines the start of the project.
(4) Each activity is assigned to exactly one period.
(5) and (6) together define the period in which an activity completes.
(7) The money that the client must pay at the end of each period is the sum of expenses of activities that finish within that period multiplied by the profit margin, β.
(8) X_{pi} is a binary variable.

If there are no progress payments, i.e., β=0, the contractor should minimize their costs; therefore, the optimal solution would be to complete as late as possible, i.e., the late start schedule is optimal. If there are no activity expenses, i.e., C_i= 0 for all i, the optimal solution would be complete as early as possible, i.e., the early start schedule is optimal. The late and early start schedules are found by the well-known Critical Path Method. For the sake of completeness, we give the details of this method.

The Critical Path Method (CPM)

In the late 1950s, Remington Rand Univac and DuPont Corporation introduced CPM to schedule project activities based on a mathematical algorithm. By using the duration and precedence relationship of activities, critical and non-critical paths are determined. The method forms the early start schedule (ESS) in the forward pass starting from the first activity in the project and the longest path named as critical path gives the earliest completion time of the project. Oppositely, starting from the deadline of the project, via using the backward pass of the CPM late start schedule (LSS) of activities are determined. The recursive equations to form these schedules are stated as follows:

There are N activities in the project.

$ES_i=Max(ES_j+d_j)$ for $\forall$ j $\in IP_i$, and $\forall$ i

$EF_i=ES_i+d_i$ for $\forall$ i

The forward pass of the CPM begins from the first activity and finishes when the early start time of the last activity N is found.

Backward pass recursive equations are given below:

LF_n=Deadline

IS_i: set of immediate successors of activity i

$LF_i=Min(LF_j-d_j)$ for $\forall$ j $\in IS_i$, and $\forall$ i

$LS_i=LF_i-Duration_i$ for all i

2.3 Complexity of the Problem

We show, through Theorem 1, that our problem that is modelled in Section 2.2, is strongly NP-hard.

Theorem 1: Our problem is strongly NP-hard.

Proof: The objective of the problem is:

$$\text{Max } \sum_{p=1}^{P} exp\left(-\alpha . A_p\right) . CF_p - \sum_{i=1}^{N} C_i . exp(-\alpha . T_i)$$

where $CF_p = \sum_{i \in p} \beta . C_i$

When β=0, the objective function reduces to

$$\text{Max } -\sum_{i=1}^{N} C_i . exp(-\alpha . T_i) \equiv \text{Min } \sum_{i=1}^{N} C_i . exp(-\alpha . T_i)$$

The problem using the above objective and ignoring the deadline constraint (taking D very big value) is equivalent to minimizing total weighted completion time problem on parallel identical machines and with precedence constraints, i.e., $P \mid prec \mid \sum_{i=1}^{N} w_i . exp(-\alpha . T_i)$ is scheduling terminology.

De Reyck and Leus (2008) showed that $1 \mid prec \mid \sum_{i=1}^{N} w_i . exp(-\alpha . T_i)$ is strongly NP-hard, so is its generalization $P \mid prec \mid \sum_{i=1}^{N} w_i . exp(-\alpha . T_i)$.

This follows, our problem residing arbitrary β and D values over the $P \mid prec \mid \sum_{i=1}^{N} w_i . exp(-\alpha . T_i)$ problem is strongly NP-hard. ■

Theorem 1 states that there can exist neither polynomial, nor pseudo-polynomial algorithm that solves our problem.

2.4 A Feasible Solution and the Optimal Solution

In this section, we present a feasible and an optimal solution of the model. The optimal solution is found by GAMS BARON nonlinear programming solver. We consider the following 8 activity precedence network shown in Figure 2.

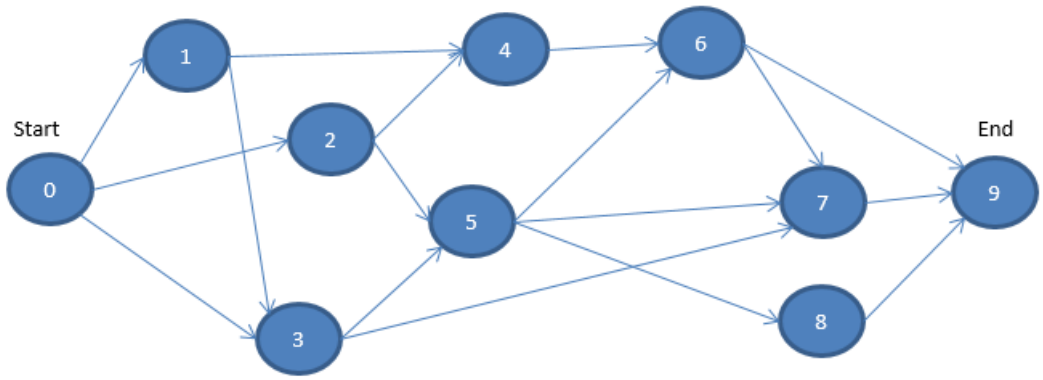

Figure 2 - An Example Network Diagram

Table 1 gives the activity durations, costs and predecessor activities.

Table 1 - An Example Problem

ID	Duration (months)	Cost ($)	Immediate Predecessors
0	0	0	-
1	3	600	0
2	2	1800	0
3	5	700	0,1
4	6	1600	1,2
5	8	2000	2,3
6	4	1500	4,5
7	3	1900	3,5,6
8	4	600	5
9	0	0	6,7,8

Let the project deadline be 30 months, and the project has 3 payment times that are on the 10^{th}, 20^{th} and 30^{th} months. The discount rate (α) is assumed to be 0.10 per year and profit margin (β) is taken as 1.20.

Table 2 gives the earliest and latest completion time of activities, found by the CPM.

Table 2 - Earliest and Latest Schedule of the Project

ID	Earliest Start Time	Earliest Finish Time	Earliest Period	Latest Start Time	Latest Finish Time	Latest Period
0	0	0	1	7	0	1
1	0	3	1	7	10	1
2	0	2	1	13	15	2
3	3	8	1	10	15	2
4	3	9	1	17	23	3
5	8	16	2	15	23	3
6	16	20	2	23	27	3
7	20	23	3	27	30	3
8	16	20	2	26	30	3
9	23	23	3	30	30	3

Note that both schedules are feasible for our problem. For the ESS schedule; Table 3 summarizes the objective function value calculations.

Table 3 - Early Start Schedule Objective Function

ID	Early Finish	$\sum_{p=1}^{P} exp(-\alpha . A_p) . CF_p$	$\sum_{i=1}^{N} C_i . exp(-\alpha . T_i)$	ESS Obj Function
0	0	0.0	0.0	0.0
1	3	662.4	585.2	77.2

Table 3 – Early Start Schedule Objecive Function (Continued)

ID	Early Finish	$\sum_{p=1}^{P} exp(-\alpha . A_p) . CF_p$	$\sum_{i=1}^{N} C_i . exp(-\alpha . T_i)$	ESS Obj Function
2	2	1987.3	1770.2	217.0
3	8	772.8	654.9	118.0
4	9	1766.5	1484.4	282.1
5	16	2031.6	1750.3	281.2
6	20	1523.7	1269.7	253.9
7	23	1775.7	1568.6	207.1
8	20	609.5	507.9	101.6
9	23	0.0	0.0	0.0
Total		11129.4	9591.2	1538.2

For the LSS schedule; Table 4 summarizes the objective function value calculations.

Table 4 - Latest Start Schedule Objective Function

ID	Late Finish	$\sum_{p=1}^{P} exp(-\alpha . A_p) . CF_p$	$\sum_{i=1}^{N} C_i . exp(-\alpha . T_i)$	LSS Obj Function
0	0	0.0	0.0	0.0
1	10	662.4	552.0	110.4
2	15	1828.4	1588.5	239.9
3	15	711.0	617.7	93.3
4	23	1495.3	1320.9	174.4
5	23	1869.1	1651.2	218.0
6	27	1401.8	1197.8	204.1
7	30	1775.7	1479.7	295.9

Table 4 - Latest Start Schedule Objective Function (Continued)

ID	Late Finish	$\sum_{p=1}^{P} exp(-\alpha . A_p) . CF_p$	$\sum_{i=1}^{N} C_i . exp(-\alpha . T_i)$	LSS Obj Function
8	30	560.7	467.3	93.5
9	30	0.0	0.0	0.0
Total		10304.5	8875.1	1429.4

As it can be seen from Table 3, the objective function value is found as 1538.2 whereas it is 1429.4 for LSS, according to Table 4. It is seen that ESS performs better than LSS for this problem instance.

Another reasonable feasible solution could be formed by using ESS to decide on the intervals of each activity. Then, LSS can be used for the completion times within each interval. Such a solution would use the advantage of ESS to receive the payments earlier and LSS to pay the costs later.

According to the ESS, without changing the periods of the activities, the activities are rescheduled as late as possible and the completion times are reported in Table 5.

Table 5 - Latest Start in Earliest Period Objective Function

ID	Finishing Time	$\sum_{p=1}^{P} exp(-\alpha . A_p) . CF_p$	$\sum_{i=1}^{N} C_i . exp(-\alpha . T_i)$	Objective Function
0	0	0	0.0	0.0
1	3	662.4	585.2	77.2
2	4	1987.3	1741.0	246.3
3	8	772.8	654.9	118.0

Table 5 - Latest Start in Earliest Period Objective Function (Continued)

ID	Finishing Time	$\sum_{p=1}^{P} exp(-\alpha . A_p) . CF_p$	$\sum_{i=1}^{N} C_i . exp(-\alpha . T_i)$	Objective Function
4	10	1766.5	1472.1	294.4
5	16	2031.6	1750.3	281.2
6	20	1523.7	1269.7	253.9
7	30	1775.7	1479.7	295.9
8	20	609.5	507.9	101.6
9	30	0.0	0.0	0.0
Total		11129.4	9460.8	1668.6

If all activities are scheduled as in Table 5, objective function of the contractor is 1668.6. The Gantt Chart of ESS and latest start schedule in earliest periods are illustrated in Figure 3.

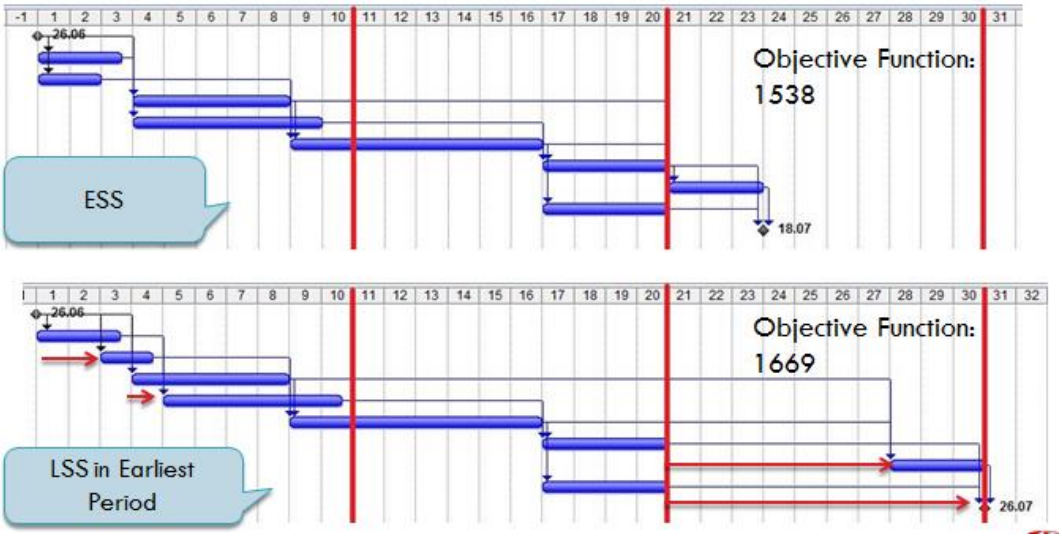

Figure 3 - The Gant Chart of ESS and LSS in Earliest Period

From Figure 3, it is seen that only the activities 2, 4, 7 and 9 are shifted forward until LSS in earliest period is obtained. Due to these shifts, the objective function improves from 1538 to 1669.

Since it is a small-sized problem, GAMS BARON solver could find the optimal solution and return the results reported in Table 6.

Table 6 – The Optimal Solution

ID	Optimal Finishing Time	Optimal Period	$\sum_{p=1}^{P} exp(-\alpha . A_p) . CF_p$	$\sum_{i=1}^{N} C_i . exp(-\alpha . T_i)$	Obj. Func.
0	0	1	0.0	0.0	0.0
1	5	1	662.4	575.5	86.9
2	10	1	1987.3	1656.1	331.2
3	10	1	772.8	644.0	128.8
4	20	2	1625.2	1354.4	270.9
5	20	2	2031.6	1693.0	338.6
6	27	3	1401.8	1197.8	204.1
7	30	3	1775.7	1479.7	295.9
8	30	3	560.7	467.3	93.5
9	30	3	0.0	0.0	0.0
Total			10817.6	9067.7	1749.

The cash flow diagram of the optimal solution is given in Figure 4.

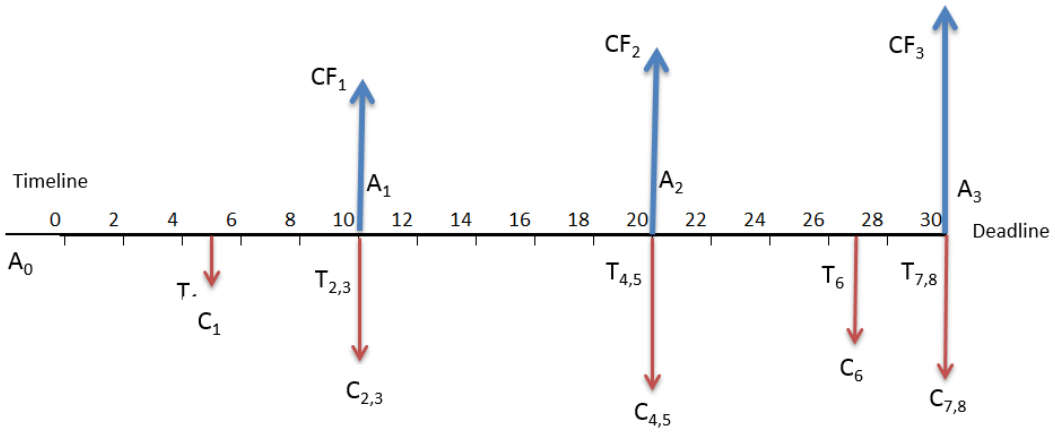

Figure 4 - The Cash Flow Diagram of the Optimal Solution

The objective function value of the optimal solution is 1749.9 whereas it was 1668.6 with latest start in earliest period schedule. When the optimal schedule is compared with the latest start in earliest period schedule, it is seen that activities 4, 6 and 8 are scheduled in their later periods. Although activity 4 could also be scheduled in period 3; in the optimal solution, it is scheduled on period 2.

CHAPTER 3

LITERATURE SURVEY

In this chapter, we review the literature on project payment scheduling problems with discounted cash flows.

We classify the research as resource constrained and unconstrained problems. We refer the reader to Herroelen et al. (1997) for different classification schemes.

3.1 Resource Constrained Problems

The studies by Smith Daniels et al. (1996), Icmeli and Erenguc (1996), De Reyck and Herroelen (1997), Sepil and Ortac (1997), Najafi and Niaki (2005), Najafi and Azimi (2009), Ritwick and Paul (2013) consider resource constrained problems without mode decisions. The studies by Özdamar and Dündar (1996), Ulusoy and Cebelli (2000), Mika et al. (2005), Seifi and Moghaddam (2008), Chen et al (2010) include mode decisions to their resource constrained problem.

We summarize the studies in chronological order.

Smith Daniels et al. (1996) consider with capital constrained project scheduling problem and offer three heuristic procedures for its solution. The objective function of their problem maximizes the net present value of cash inflows, outflows and capital costs whereas capital is counted as a renewable resource. With an initial capital availability, cash inflows and outflows affect the capital throughout the project. They test the performance of optimization guided

heuristics and conclude that two revised optimization guided heuristics performed better than a randomly derived bound and cash flow weight heuristic procedure.

Özdamar and Dündar (1996) investigate probabilistic cash inflows for a multi-mode capital constrained project scheduling problem. They consider the construction as focusing on the case of selling the flats while the construction is ongoing. Thus, they included activity modes, which include time and cost tradeoffs. The capital constrained model of Herroelen et al (1995) is used including both renewable and non-renewable resources are included in the problem. Their objective function is to maximize the net present value of the project which is reduced by penalty of tardiness and opportunity cost of initial capital. An adaptive scheduling program is proposed to solve the problem and they also emphasize the program as a useful simulation tool.

Icmeli and Erenguc (1996) propose a depth-first branch and bound algorithm for the resource constrained project payment scheduling problem. They assume that each activity has either positive or negative cash flow and aim to maximize net present value of these cash flows. In their model, only non-renewable resources are considered. Their branching is done according to the complete schedule that ignores resource feasibility. They first search for a schedule feasible solution and for the nodes that are resource infeasible but schedule feasible they further branch until they reach resource feasibility. While maintaining schedule feasibility, a set of solutions with a tolerance factor of at most 5% of the optimal solution is represented for comparison.

De Reyck and Herroelen (1997) offer a dept-first Branch and Bound algorithm for the resource constrained project scheduling problem with discounted cash flows and generalized precedence relations. They include renewable resource in their problem as constraints. They do not consider any payment model and assumes negative or positive cash flow occur at activity completions. The concept of

minimal delaying modes are introduced to get away the resource conflicts. Their computational study is presented up to 50 activities and important parameters are selected to see their effects on the problem difficulty.

Sepil and Ortac (1997) investigate the performance of three heuristic procedures for resource constrained projects with progress payments. They assumed that cash inflows occur periodically and cash outflows occur at the activity completion times. Performance of single net present value comparison rule, pairwise present value comparison rule, and activity profit curve sloper rule are compared with each other. Their results show that the three heuristics provide near-optimal schedules with respect to net present value maximization without delaying the deadline extensively.

Ulusoy and Cebelli (2000) use event completion payment model in their multi-mode resource constrained project scheduling problem. They consider renewable resources and time-resource trade-offs. In their problem, the amount and timing of the payments are determined so as to minimize the absolute percent deviation of the ideal net present values of the contractor and client. The proposed double loop genetic algorithm performs in reasonable times with equitable solutions.

Ulusoy et al. (2001) employ a genetic algorithm for multi-mode resource constrained project scheduling problem for four payment models. They consider renewable and non-renewable resources. The objective is to maximize the net present value of positive and negative cash flows. They consider time-cost and/or time-resource tradeoffs. For different discount rates and profit margins, genetic algorithm and local constraint based analysis are used to solve the problem and it is found that genetic algorithm outperforms comparing the other methods. They conclude that when profit margin increases, the progress payment model tends to schedule activities that have relatively higher costs earlier. Furthermore, as

discount rate increases, the progress payment model starts to behave like lump sum payment model as tending to decrease the project completion time.

Mika et al. (2005) consider four payment models for multi-mode resource constrained project scheduling problem. They consider both renewable and nonrenewable resources and make time and resource trade-off. Lump sum payments, payments at activities' completion times, equal time intervals and progress payment methods are used in their study. They aim to maximize the net present value of all cash flows. They present simulated annealing and tabu search approaches and discuss the results on different payment models, different frequency of payments and different discount rates. They find that for smaller number of activities, the performance of tabu search algorithm performs better while for large number of activities the performance of simulated annealing is better.

Najafi and Niaki (2005) propose a heuristic method to solve resource investment problems with discounted cash flows. They use the payment model of payments at predefined event occurrences. Renewable resources are included in the problem in constraints and objective function due to the nature of resource investment problem. Their objective function maximizes the net present value of the cash flows which include project costs including resource costs and payments made during the project. Their heuristic procedure are compared with the best solutions obtained from mathematical model and found that while solutions are very close, CPU time for the heuristics is much less than CPU time of mathematical model.

Seifi and Moghaddam (2008) analyze four payment models for multi-mode resource constrained project scheduling problem. In their study, time-resource and cost tradeoffs are made and both renewable and nonrenewable resources are considered. They present a bi-objective model which maximizes the net present value and minimizes the holding cost of activities. A mathematical model is

verified by small sized problems whereas simulated annealing is used for various sizes of problems with different payment models. Results on different payment models with varying activity sizes, discount values, holding cost rate and number of payment times are compared at the end of the paper. They present the results that with the increase in discount rate and holding cost rate, the net present value decreases and with the increase of period size, the objective function value increases for progress payment model.

Najafi and Azimi (2009) study resource investment project scheduling problem with discounted cash flows with the extension of tardiness penalties. They include the delay penalty to the problem Najafi and Niaki (2005) study. A priority rule based heuristic is used to solve the problem and comparison between mathematical model and heuristic solutions are presented.

Chen et al (2010) study on an ant colony optimization approach for multimode resource-constrained project scheduling problem with discounted cash flows. Their model maximizes the net present value of all cash flows including payments, expenses, and bonus-penalty. They include both renewable and nonrenewable resources in their model. They assume that expenses of the contractor are at event completions, payment are at event occurrences and the number of payments are given. Activity on Arc network precedence is converted into a construction graph (MoN graph) to make the data more compatible with the solution approach. They compare their ant colony optimization approach with genetic algorithm, simulated annealing and tabu search solutions, and find that their ant colony optimization approach outperforms the other approaches.

Ritwick and Paul (2013) use the features of particle swarm optimization to solve resource constrained project scheduling problem with discounted cash flows. They assume that cash flows are deterministic but uniformly distributed between a negative and positive value, thus their problem does not include a payment model.

They claim that although for networks more than 40 activities the algorithm does not find an optimum solution in reasonable times, its near optimum solutions are satisfying considering the running time.

3.2 Unconstrained Problems

The studies by Russell (1970), Grinold (1972), Erenguc et al. (1993), De Reyck and Herrolen (1996), Kazaz and Sepil (1996), Dayanand and Padman (1997), He and Xu (2006), Vanhoucke et al. (2003), He et al. (2009) consider project scheduling problems without resource constraints.

We review the studies on unconstrained problems in chronological order.

Russell (1970) introduces the problem of maximizing the net present value of the cash flows in a project. He uses an event based model that both considers positive and negative cash flows during the project. He presents a mathematical model and offers a modified Ford and Fulkerson (1961)'s out of kilter algorithm as finding the costs of flows through the arcs for large size network problems.

Grinold (1972) transforms the nonlinear program of Russell into a linear program and suggests an efficient procedure for its solution. He studies the fixed deadline problem and the trade-off curve problem for the net present value and project duration criteria.

Erenguc et al. (1993) consider the project scheduling problem with time/cost trade-offs. They assume that the payments are received at the completions of the defined events. They decide on the activity modes and the project schedule so as to maximize positive and negative cash flows. Their algorithm uses the generalized Benders decomposition idea and their computational results reveal that the instances with up to 64 activities can be solved.

De Reyck and Herrolen (1996) consider the unconstrained maximum net present value project scheduling problem and propose a Branch and Bound algorithm that uses arbitrary minimal and maximal time lags between the start and completion of the activities. They do not consider any payment model and assume that cash flow of activities are either positive or negative. Their solution procedure solves the problems up to 100 activities in reasonable times and they offer the procedure for the calculation of upper bounds for the resource constrained npv problem, which they will actually use it in another paper.

Kazaz and Sepil (1996) and Vanhoucke et al. (2003) consider a project scheduling problem with discounted cash flows and progress payments. Their objective is to maximize the net present value of the cash flows. They assume the receipts are made at the end of the specified time periods to cover the expenses made during the period. The expenses are assumed to be proportional with the amount of work performed. Kazaz and Sepil (1996) give a mixed integer linear programming formulation of the problem and report favorable results on its solution times. Vanhoucke et al. (2003) transform the problem into a weighted earliness–tardiness project scheduling problem and propose a branch-and-bound algorithm for its optimal solution. Their computational results reveal that the branch-and-bound algorithm is capable of solving problems with upto 50 activities.

Dayanand and Padman (1997) examine several project payment scheduling models. The models find the amount and time of the progress payment and they are basically of two types: event based or activity based. All models aim to maximize net present worth of all cash flows and they differ by their problem representations and assumptions. They mention the models are mixed integer linear or nonlinear programs and hence they are intractable for large projects. They demonstrate the solutions of the models on an example problem instance.

Multimode project payment scheduling problem with bonus-penalty structure is studied by He and Xu (2006). In their problem, timing of the events and payments, amount of the payments and activity modes are decision variables whereas the number of payments are predefined. The contractor optimization model and the client optimization models are presented. Two modules simulated annealing is used to solve the problems and effects of the bonus-penalty structures on the flexibility of the project payment schedules are discussed.

He et al. (2009) consider multimode project scheduling problem for a fixed number of payments subject to a specific deadline. The problem is to select the activity modes and schedule the activities so as to maximize the net present value of the contractor. They consider the four payment model which are defined as lump-sum, payments at event occurrences, equal time intervals and progress payments. Simulated annealing, tabu search, multi-start iterative improvement and random sampling solution methods are compared with each other for different values of number of payments, interest rate and profit margin of the contractor. Simulated annealing is found to be the best especially for large sized problem instances.

The most closely studies to ours are due to Kazaz and Sepil (1996) and Vanhoucke et al. (2003). Both studies assume that the activity costs are incurred at the end of the payment periods up to the progress of the activity. We assume all cost incurred by the activity is charged when the activity is complete.

CHAPTER 4

SOLUTION APPROACH

Recall that our problem is strongly NP-hard. To find exact solutions, once could use the mixed integer non-linear problem introduced in Chapter 2. However, attributing the complexity of the problem, our model is likely to full into computation troubles. An alternative to the model is to use an implicit enumeration technique like a branch and bound algorithm. In this study, we present a branch and bound algorithm with the hope of solving the problem in reasonable times. In the following section, properties of an optimal solution that forms our branching scheme, the method of finding the upper and lower bounds and branching scheme are discussed in detail.

4.1 Properties of an Optimal Solution

From the contractor view point, if the activities that will be finished in a period are definite, then it is more profitable for contractor to finish the activities as late as possible in the defined period. Since the contractor receives the payment at the end of period, and expenses incur whenever an activity finishes, the contractor will reduce the net present value of the expenses by scheduling the activities as late as possible.

Theorem 2 states this result formally. To state Theorem 2, following notation is needed:

A_p=end of period p

A_p-A_{p-1}=length of period p

S_p = set of activities that complete in period p

Theorem 2: In an optimal solution, the tasks in S_p are sequenced by late start schedule with deadline value of A_p.

Proof: Recall that the first part of the objective function is:

$$\sum_{p=1}^{P} exp(-\alpha. A_p). CF_p = \sum_{p=1}^{P} \sum_{i \in S_p} \beta * C_i. exp(-\alpha. A_p) \quad (4.1)$$

(4.1) Is irrelevant of optimization given set S_ps.

The second part of the objective function is

$$\sum_{i=1}^{N} C_i. \exp(-\alpha. T_i) \quad (4.2)$$

(4.2) is minimized when T_is are maximized and T_is are maximized when LSS is used.

As (4.1) is irrelevant of optimization and (4.2) is minimized by LSS, the overall objective ((4.1)-(4.2)) is maximized by LSS. ■

We define our branching scheme by using the result of Theorem 1.

4.2 Upper Bound

In this section, we present two upper bounding procedures, which are proposed to enhance the efficiency of our branch and bound algorithm.

We use the following notation to state our upper bounds:

A_{EC_i}= Period finishing time for activity *i* if ESS is used

A_i^*= Period finishing time for activity *i* for the optimal solution

T_{HC_i}= Completion time of activity *i* such that

If $A_{EC_i} < A_{LC_i}$ then $T_{HC_i} = A_{EC_i}$

If $A_{EC_i} = A_{LC_i}$ then activity *i* definitely finishes within period EC_i, and T_{HC_i} is the latest finishing time between the activities those also definitely finish within period EC_i.

T_{LC_i}= Completion time of activity *i* when LSS is used

T_i^*= Completion time of activity *i* for the optimal solution

$UB_1 = \sum_{i=1}^{N} \beta * C_i . exp(-\alpha . A_{EC_i}) - \sum_{i=1}^{N} C_i . exp(-\alpha . T_{LC_i})$

$UB_2 = \sum_{i=1}^{N} \beta * C_i . exp(-\alpha . A_{EC_i}) - \sum_{i=1}^{N} C_i . \exp(-\alpha . T_{HC_i})$

Z^*= optimal objective function value

$= \sum_{i=1}^{N} \beta * C_i . exp(-\alpha . A_i^*) - \sum_{i=1}^{N} C_i . exp(-\alpha . T_i^*)$

Theorem 3 and Theorem 4 state the validity of the upper bounds.

Theorem 3: UB_1 is a valid upper bound on Z^*.

Proof: As $A_{EC_i} \leq A_i^*$

$$\sum_{i=1}^{N} \beta * C_i . exp\left(-\alpha . \mathrm{A_{EC_i}}\right) \geq \sum_{i=1}^{N} \beta * C_i . exp(-\alpha . {\mathrm{A}_i}^*) \tag{4.3}$$

$$-\sum_{i=1}^{N} C_i . \exp\left(-\alpha . T_{LC_i}\right) \geq -\sum_{i=1}^{N} C_i . exp(-\alpha . {T_i}^*) \tag{4.4}$$

As $T_{LC_i} \geq T_i$

Add (4.3) and (4.4) and get (4.5).

$$\mathrm{UB_1} = \sum_{i=1}^{N} \beta * C_i . exp\left(-\alpha . \mathrm{A_{EC_i}}\right) - \sum_{i=1}^{N} C_i . \exp(-\alpha . T_{LC_i}) \geq$$

$$\sum_{i=1}^{N} \beta * C_i . exp(-\alpha . {\mathrm{A}_i}^*) - \sum_{i=1}^{N} C_i . \exp(-\alpha . {T_i}^*) = \mathrm{Z}^* \blacksquare \tag{4.5}$$

Theorem 4: $\mathrm{UB_2}$ is a valid upper bound on Z^*.

Proof: Let,

$\mathrm{x_i} = \mathrm{A_{EC_i}} - T_{HC_i}$

$\mathrm{y_i} = {\mathrm{A}_i}^* - {T_i}^*$

There are two possibilities for $\mathrm{A_{EC_i}}$ and ${\mathrm{A}_i}^*$:

1) $\mathrm{A_{EC_i}} = {\mathrm{A}_i}^*$ then $T_{HC_i} \geq {T_i}^*$ (by the definition of T_{HC_i}); therefore,
   $\mathrm{x_i} = \mathrm{A_{EC_i}} - T_{HC_i} \leq {\mathrm{A}_i}^* - {T_i}^* = \mathrm{y_i}$
2) $\mathrm{A_{EC_i}} < {\mathrm{A}_i}^*$ then $T_{HC_i} = \mathrm{A_{EC_i}}$ (by the definition of T_{HC_i}); therefore,
   $\mathrm{x_i} = \mathrm{A_{EC_i}} - T_{HC_i} = 0 \leq {\mathrm{A}_i}^* - {T_i}^* = \mathrm{y_i}$

Thus, $\mathrm{x_i} \leq \mathrm{y_i}$

Letting $T_{HC_i} = \mathrm{A_{EC_i}} - \mathrm{x_i}$, $\mathrm{UB_2}$ can be rewritten as

$$\mathrm{UB_2} = \sum_{i=1}^{N} \beta * C_i . exp\left(-\alpha . \mathrm{A_{EC_i}}\right) - \sum_{i=1}^{N} C_i . \exp(-\alpha . (\mathrm{A_{EC_i}} - \mathrm{x_i}))$$

$= \sum_{i=1}^{N} \beta * C_i . exp\left(-\alpha . A_{EC_i}\right) - \sum_{i=1}^{N} C_i . \exp(-\alpha . A_{EC_i}) . \exp(\alpha . x_i))$

$= \sum_{i=1}^{N} C_i . (\beta - \exp(\alpha . x_i)) . exp\left(-\alpha . A_{EC_i}\right)$ (4.6)

Letting T_i^*=A_i^*- y_i, Z^* can be rewritten as

$Z^* = \sum_{i=1}^{N} \beta * C_i . exp(-\alpha . A_i^*) - \sum_{i=1}^{N} C_i . \exp(-\alpha . (A_i^* - y_i))$

$= \sum_{i=1}^{N} \beta * C_i . exp(-\alpha . A_i^*) - \sum_{i=1}^{N} C_i . \exp(-\alpha . A_i^*) . exp(\alpha . y_i)$

$= \sum_{i=1}^{N} C_i . \exp(-\alpha . A_i^*) . (\beta - exp(\alpha . y_i))$ (4.7)

$x_i \leq y_i$ follows

$\exp(\alpha . x_i) \leq \exp(\alpha . y_i) \rightarrow (\beta - \exp(\alpha . x_i)) \geq (\beta - \exp(\alpha . y_i))$ (4.8)

(4.8) follows

$\sum_{i=1}^{N} \beta * C_i . exp\left(-\alpha . A_{EC_i}\right) - \sum_{i=1}^{N} C_i . \exp(-\alpha . T_{HC_i}) \geq$

$\sum_{i=1}^{N} \beta * C_i . exp(-\alpha . A_i^*) - \sum_{i=1}^{N} C_i . exp(-\alpha . T_i^*)$ (4.9)

As $A_{EC_i} \leq A_i^* \rightarrow exp\left(-\alpha . A_{EC_i}\right) \geq exp(-\alpha . A_i^*)$

From (4.6), (4.7), and (4.9), one can write

$UB_2 \geq Z^*$ ■

Theorem 5 states UB_2 is more powerful than UB_1, i.e., $UB_1 \geq UB_2$.

Theorem 5: UB_2 dominates UB_1.

Proof: $-\sum_{i=1}^{N} C_i.\exp\left(-\alpha.T_{LC_i}\right) \geq -\sum_{i=1}^{N} C_i.\exp(-\alpha.\mathrm{T_{HC_i}})$

(4.10)

As $T_{LC_i} \geq \mathrm{T_{HC_i}}$

Add $\sum_{i=1}^{N} \beta * C_i.exp\left(-\alpha.\mathrm{A_{EC_i}}\right)$ to both sides of (4.10) and get (4.11).

$$UB_1 = \sum_{i=1}^{N} C_i.\left(\beta * exp\left(-\alpha.\mathrm{A_{EC_i}}\right) - exp\left(-\alpha.T_{LC_i}\right)\right) \geq$$
$$\sum_{i=1}^{N} \beta * C_i.exp\left(-\alpha.\mathrm{A_{EC_i}}\right) - \sum_{i=1}^{N} C_i.\exp(-\alpha.\mathrm{T_{HC_i}}) = UB_2$$

(4.11)

(4.11) is equivalent to

$UB_1 \geq UB_2$ ■

We illustrate the UB_1 and UB_2 computations in Table 7 via the following example problem.

Table 7 - Calculation of Upper Bound

m	A_{EC_m}	T_{LC_m}	EC_m	LC_m	UB_1	UB_2
0	10	10	1	1	=0*1.2*EXP(-0.1*10/12)-0*EXP(-0.1*10/12)=0	=1.20*0*EXP(-0.10* 10 /12)-0*EXP(-0.10* 10 /12) = 0
1	10	10	1	1	=600*1.2*EXP(-0.1*10/12)-600*EXP(-0.1*10/12)=110.4	=1.20*600*EXP(-0.10* 10 /12)-600*EXP(-0.10* 10 /12) = 110.4
2	10	15	1	2	=1800*1.2*EXP(-0.1*10/12)-1800*EXP(-0.1*15/12)=398.8	=1.20*1800*EXP(-0.10* 10 /12)-1800*EXP(-0.10* 10 /12) = 331.2
3	10	15	1	2	=700*1.2*EXP(-0.1*10/12)-700*EXP(-0.1*15/12)=155.1	=1.20*700*EXP(-0.10* 10 /12)-700*EXP(-0.10* 10 /12) = 128.8
4	10	23	1	3	=1600*1.2*EXP(-0.1*10/12)-1600*EXP(-0.1*23/12)=445.6	=1.20*1600*EXP(-0.10* 10 /12)-1600*EXP(-0.10* 10 /12) = 294.4
5	20	23	2	3	=2000*1.2*EXP(-0.1*20/12)-2000*EXP(-0.1*23/12)=380.4	=1.20*2000*EXP(-0.10* 20 /12)-2000*EXP(-0.10* 20 /12)= 338.6
6	20	27	2	3	=1500*1.2*EXP(-0.1*20/12)-1500*EXP(-0.1*27/12)=325.9	=1.20*1500*EXP(-0.10* 20 /12)-1500*EXP(-0.10* 20 /12) = 253.9
7	30	30	3	3	=1900*1.2*EXP(-0.1*30/12)-1900*EXP(-0.1*30/12)=295.9	=1.20*1900*EXP(-0.10* 30 /12)-1900*EXP(-0.10* 30 /12) = 295.9
8	20	30	2	3	=600*1.2*EXP(-0.1*20/12)-600*EXP(-0.1*30/12)=142.2	=1.20*600*EXP(-0.10* 20 /12)-600*EXP(-0.10* 20 /12) = 101.6
9	30	30	3	3	=0*1.2*EXP(-0.1*30/12)-0*EXP(-0.1*30/12)=0	=1.20*0*EXP(-0.10* 30 /12)-1.20*0*EXP(-0.10* 30 /12) = 0
Total					2254.2	1854.9

Recall that the effort spent to find UB_2 is similar to that of UB_1. Theorem 5 states that UB_2 has higher quality. Hence, we use UB_2 in our branch and bound algorithm. We hereafter refer to UB_2 simply as UB.

4.3 Lower Bound

We use the following procedure to find an initial feasible solution to our branch and bound algorithm. The procedure considers two phases.

Phase 1. Construction

Find the period in which activity *i* could complete earliest, i.e.,

$$A_{EC_i-1} < T_{EC_i} \leq A_{EC_i}$$

S_p = set of activities that complete in period p, i.e. $(A_{r-1}, A_r]$

Starting from period *P*, schedule the jobs in S_p, according to the late start schedule. We call the schedule as Latest Start in Earliest Period (LS-EP) schedule.

Phase 2. Improvement

The schedule obtained from Construction phase is fed to the improvement phase. We start from period *P*-1 and look for the chance of moving activities which are initially scheduled to the current period to the next period. We stop when the first period is reached.

A move is defined as shifting task from period r to period r+1. Best move is the one that improves the objective function value by the maximum amount. The resulting solution is worse when the maximum amount is negative.

We employ following two strategies to improve the schedule found in the construction phase:

Strategy 1.

Perform two iterations for each period. In each iteration, select the best move. The best move in an iteration may worsen the objective function value but in later iterations the move may create some space to improve the objective function value.

Strategy 2.

Perform the moves for each period till the objective function value worsens as a second time.

We apply both strategies to our initial solution and select the strategy that leads largest objective function value. We let the resulting solution be LB_H of the problem.

4.4 Branching Scheme

We start scheduling from the last period and stop when the first period is reached. At level 0, we define on the last period's schedule. At the beginning of each period, we first fix a set of activities whose earliest completion times are in the current period p, i.e. $T_{EC_i} \in (A_{p-1}, A_p]$. After fixing those activities, we open two types of nodes.

Type 1 nodes

Close period *P* and proceed to period *P*-1

Type 2 nodes

Add an eligible activity to period *P*

We say an activity is eligible for assignment if all of its successors are already scheduled.

We index the activities such that $i<j$ implies that activity i is not successor of activity j. To avoid the duplication of the solutions, while adding an activity to the current period, we only consider the activities having lower indices. In other words, if a tree has two nodes that has activity indices of j and j-1 which are branched from the same node, the node that has activity j-1 implies activity j is going to be scheduled in an earlier period.

At level r, we consider period P-r+1 and fix the activities whose earliest completion times are in $(A_{r-1}, A_r]$. Note that eligible activities are scheduled to the periods larger than P-r with the preceding nodes. After fixing for each remaining activity, for the node that considers the addition of activity i to partial schedule S, we calculate an upper bound by extending UB to the partial schedule as follows.

Let,

$S' = S \cup \{i\}$

$Z(S')$ = Net Present Worth of activities in S'

NS: set of unscheduled activities

$UB(S' \cup NS) = Z(S') + UB(NS)$

Earliest periods of activities do not change; however, their latest completion times may change as some activities in S might have been scheduled to earlier periods that have not been reached yet.

$UB(NS) = \sum_{i \notin S} \beta * C_i . exp(-\alpha . A_{EC_i}) - \sum_{i \notin S} C_i . exp(-\alpha . T_{HC_i})$

We terminate further branching from the node when

$UB(S' \cup NS) < LB_{BEST}$

where LB_{BEST} is the current best known feasible solution. We continue branching with the highest UB of the nodes when all possible activities are branched by the ancestor node.

We use depth first strategy, i.e., always go to the depth of the tree and backtrack only when there is no feasible or promising node to explore further. We select this strategy due to its relatively low memory requirements.

We stop when we backtrack to level 0. We start with $LB_{BEST}= LB_H$ and update LB_{BEST} whenever a complete solution with a better objective function value is found.

We illustrate our branching scheme in Figure 5 whose network is depicted in Figure 2.

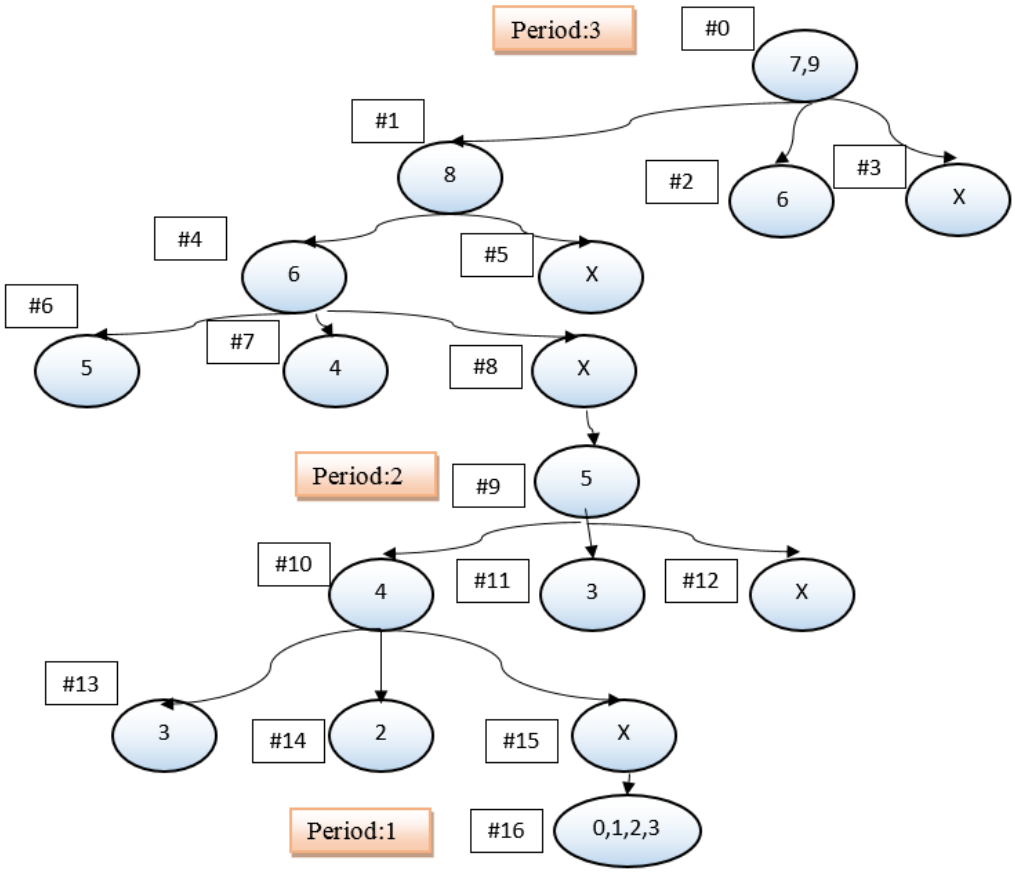

Figure 5 - Branch and Bound Tree Structure

The information conveyed on each node is reported in Table 8.

Table 8 – The Parameters of Nodes in the Branch and Bound Tree

Node ID	Node Information	Scheduling Period	UB	Branching
0	Act. 8,10 are definitely in period 3 LB_H=1669	3	1855	Branched
1	Act. 8,9,10 are in period 3	3	1847	Branched as UB_1>UB_2 UB_1>UB_3
2	Act. 7,8,10 are in period 3, 9 is not in period 3	3	1676	
3	Only act. 8,10 are in period 3	3	1726	
4	Act. 7,8,9,10 are in period 3	3	1797	Branched as UB_4>UB_5
5	Only act. 8,9,10 are in period 3	3	1718	
6	Act. 6,7,8,9,10 are in period 3	3	1676	
7	Act. 5,7,8,9,10 are in period 3, 6 is not in period 3	3	1663	Fathomed as UB_7>LB_H
8	Only act. 7,8,9,10 are in period 3	3	1783	Branched as UB_8>UB_6 UB_8>UB_7
9	Act. 7,8,9,10 are in period 3, 6 is definitely in period 2	2	1783	Branched as single node
10	Act. 7,8,9,10 are in period 3, 5,6 are in period 2	2	1759	Branched as UB_{10}>UB_{11} UB_{10}>UB_{12}
11	Act. 7,8,9,10 are in period 3, 4,6 are in period 2, 5 is not in period	2	1633	

Table 8 – The Parameters of Nodes in the Branch and Bound Tree (Continued)

Node ID	Node Information	Scheduling Period	UB	Branching
12	Act. 7,8,9,10 are in period 3, only 6 is in period 2	2	1684	
13	Act. 7,8,9,10 are in period 3, 4,5,6 are in period 2	2	1708	
14	Act. 7,8,9,10 are in period 3, 3,5,6 are in period 2, 4 is not in	2	1618	
15	Act. 7,8,9,10 are in period 3, only 5,6 are in period 2	2	1750	Branched as $UB_{15}>UB_{13}$ $UB_{15}>UB_{14}$
16	Act. 7,8,9,10 are in period 3, 5,6 are in period 2, 1,2,3,4 are in period 1, LB is updated as 1750	1	1750	Terminal node

As it can be seen from Figure 5, node 8 and node 15 are Type 1 nodes whereas other nodes are Type 2.

After the tree has a complete schedule at node 16, lower bound is updated as 1749.9. The nodes are eliminated in order nodes 13,14,..,0 as all associated upper bounds are no more than the best lower bound value. As node 0 is also fathomed, the lower bound schedule found at node 16 is optimal.

Detailed calculation of ESS and LSS of the selected nodes 0, 8 and 16 are given in Appendix A.

CHAPTER 5

COMPUTATIONAL EXPERIMENT

In this chapter, we first discuss the data generation scheme. Next, we give the results of our preliminary experiment. Finally, we discuss the results of our main experiment.

5.1 Data Generation

This part explains how the problems are generated, which parameters and distributions are used and in which configurations of the problems are solved. The following parameters are to be generated for each instance: network complexity, duration of activities, cost of activities, discount rate, profit margin, deadline multiplier factor and number of periods.

To generate networks and activity durations, we use the generator called ProGen developed by Kolisch et al. (1992) via ignoring its resource and mode options. Each network has a super-source and a super-sink node that has zero duration and cost. Super-source node does not have a predecessor and it builds the network by its successors. Oppositely, super-sink node does not have a successor and one can reach all activities by going back from sink node. Excluding super-source and super-sink node from network, the networks with 35, 40, 45, 50 and 55 activities are created. For 30 activities of network, PSPLIB – A project Scheduling Problem Library by Kolisch and Sprecher (1996) is used. We create 10 instances of each network.

Kolisch et al. (1992) define the network complexity used in ProGen as the average number of non-redundant arcs per node including the super-source and super-sink node. For the network complexity 1.5 for a 35 activities has 56 successors likewise for a 50 activities network has 78 successors. In our problems, we use the network complexity as 1.5 and 2.

Using PROGEN, we generate the durations of the activities from discrete uniform distribution in [1, 10] and [1, 20]. We use two sets of cost generation methods. In the first set, the activity times and costs are proportional. The cost of an activity is set as 100 times its duration. The second set, the costs are taken from a discrete uniform distribution in [1, 20] multiplying it with 100.

The discount rate is set to 5% and 10% as yearly compatible to today's real-world settings. Since discount rate is on yearly basis, the duration is assumed as monthly basis. In other words, for unit of time, discount rate 10% is assumed to be 0.10/12 that is 0.0833, per unit time.

Profit margin is set to 1.2 and 1.3, i.e., at the end of a period, for the profit margin 1.2 the cost of all activities that have finished within the period will be received with a 20% profit without taking the time value of money into account.

We set the deadline value to M*minimum project completion time. In our experiments, we try M=1, 1.1, and 1.2, for tight, medium and loose deadline environments, respectively.

We use 5,6,7 intervals from 30 instances to 55 instances. We assume the interval lengths are equally-sized.

5.2 Preliminary Experiments

The aim of our preliminary experiment is to define the discount rate, profit margin and network complexity to be used in our main experiment. Furthermore, we test the effects of duration and cost distributions. To see the effects of our bounds, the branch and bound algorithm is run without UB and without LB and the respective results are also discussed.

In the following analysis of profit margin, discount rate, network complexity, cost and duration, tables report the average and maximum CPU seconds and total nodes of branch and bound algorithm opened for each problem combination. In the following tables, M shows the deadline multiplier. For each problem combination, 10 problems are solved.

Unless it is stated differently, the problems use discount rate as 0.10 per year, profit margin rate as 1.2, network complexity as 1.5. The duration of the activities are generated as discrete uniform distribution in [1, 10] and cost of the activities are set as 100 times of the duration of activities.

Profit Margin

To see the effect of profit margin, we try two values: 1.2 and 1.3. 80 problems are solved in different configurations and their results are presented in Table 9.

Table 9 – The Effect of the Profit Margin

				Profit Margin			
				1.2		1.3	
N	P	M		CPU	Nodes	CPU	Nodes
30	5	1.1	Average	0.04	4953	0.03	3097
			Max	0.11	13855	0.09	9106

Table 9 – The Effect of the Profit Margin (Continued)

				Profit Margin			
				1.2		1.3	
N	P	M		CPU	Nodes	CPU	Nodes
30	5	1.2	Average	0.46	37582	0.29	23745
			Max	1.66	138031	0.97	81874
30	6	1.2	Average	0.39	52674	0.17	23214
			Max	1.03	138933	0.47	61205
35	6	1.1	Average	0.62	59041	0.52	51232
			Max	2.11	180622	2.26	204097
35	6	1.2	Average	24.38	1876164	11.17	1011075
			Max	157.89	12082629	79.51	7607042
40	5	1.1	Average	7.61	654733	2.95	270934
			Max	45.65	3933912	11.59	1094943
40	6	1.1	Average	9.28	697400	4.07	357762
			Max	32.20	2660936	11.29	1212896
40	5	1.2	Average	55.28	3580440	17.61	1344469
			Max	267.06	16256733	79.64	6860284

As can be observed from Table 9, the profit margin factor has a significant effect on the problem difficulty. The complexity of the solutions increases with the decrease in the profit margin values. This is due to the fact that as profit margin decreases, the payments decrease and approach to the cost of activities which makes the objective function value less sensitive to the scheduling times. This results into trying many schedules to find the optimal solution and makes the problem hard to solve. Note from the table that when profit margin is 1.2, for N=40, P=5 and M=1.2 the average and maximum CPU times are 55.28 and 267.06 seconds respectively, whereas they are 17.61 and 79.64 seconds when the profit margin is 1.3. The average and maximum number of nodes for profit margin 1.2 are 3580440 and 16256733 respectively while they are 1344469 and 6860284 for profit margin 1.3. In our main experiments, we select the harder instances, i.e., use the profit margin of 1.2.

Discount rate

To see the effect of the discount rates, two values of α: 0.05 and 0.10 yearly are selected. 80 problems are solved and the results are presented in the table below.

Table 10 – The Effect of the Discount Rate

				DISCOUNT RATE			
				0.05		0.10	
N	P	M		CPU	Nodes	CPU	Nodes
30	5	1.1	Average	0.03	3827	0.04	4953
			Max	0.08	10140	0.11	13855
30	5	1.2	Average	0.34	27032	0.46	37582
			Max	1.22	100643	1.66	138031
30	6	1.2	Average	0.30	39424	0.39	52674
			Max	0.86	108428	1.03	138933
35	6	1.1	Average	0.46	43723	0.62	59041
			Max	1.54	136467	2.11	180622
35	6	1.2	Average	12.91	1034866	24.38	1876164
			Max	81.96	6889839	157.89	12082629
40	5	1.1	Average	4.19	379398	7.61	654733
			Max	23.62	2192227	45.65	3933912
40	6	1.1	Average	5.02	453578	9.28	697400
			Max	18.81	1997853	32.20	2660936
40	5	1.2	Average	26.66	1821443	55.28	3580440
			Max	115.47	7629199	267.06	16256733

Table 10 reports the average and maximum solution times of the branch and bound algorithm and the number of nodes for 10 problems. Note from the table that when discount factor is 0.05, N=40, P=5 and M=1.2, the average and maximum CPU times are 26.66 and 115.47 seconds respectively whereas they are 55.28 and 267.06 seconds when the discount rate is 0.10. The average and maximum number of nodes for discount rate 0.05 are 1821443 and 7629199 respectively, while they are 3580440 and 16256733 for discount rate 0.10. This is

due to the fact that as discount rate increases, time value of money increases which results into the increase in the variance of objective function value for different schedules and this makes harder to find a solution. Therefore, according to the Table 10, we can say that as the discount rate increases, the complexity of the problem increases with the CPU times and total node number.

Network Complexity

Network complexity is another important parameter for defining the network. We tested the network complexity 1.5 and 2 with activity sizes of 40, 45, and 50, period number of 5 and 6 and deadline multiplier factor of 1 and 1.1 for 70 problem instances and report the results in the following table.

Table 11 - The Effect of the Network Complexity

				Network Complexity			
				1.5		2	
N	P	M		CPU	Nodes	CPU	Nodes
40	6	1	Average	0.26	28495	0.05	4933
			Max	1.08	116463	0.11	9533
40	5	1.1	Average	7.61	654733	0.88	90095
			Max	45.65	3933912	1.87	186670
40	6	1.1	Average	9.28	697400	3.81	311761
			Max	32.20	2660936	9.95	848460
45	6	1	Average	0.80	66955	0.22	22166
			Max	3.29	303130	0.34	36426
45	5	1.1	Average	42.45	2633124	31.79	2276599
			Max	308.94	19393268	90.26	6585136
45	6	1.1	Average	119.11	7670742	95.27	5704567
			Max	323.39	21326518	219.73	13586028
50	6	1	Average	11.96	665025	8.47	722426
			Max	34.41	1998490	22.78	2080515

As can be seen from Table 11, as network complexity increases the average and maximum CPU seconds and number of nodes decrease. As the network complexity increases the number of predecessors increase in the network which in

turn decrease the time buffer of activities as the activities can be scheduled in fewer periods. This results in less reduced choices, hence easier problems. In our experiments we select the harder combination, hence set the network complexity to 1.5. For instance, for network complexity 1.5 and n=45, P=5 and M=1.1 the average and maximum CPU times are 42.45 and 308.94 seconds and the average whereas they are 31.79 and 90.26 seconds when the network complexity is 2 for the same problem. The average and maximum number of nodes also decrease from 2633124 to 2276599 nodes and from 19393268 to 6585136 nodes, when the network complexity increase from 1.5 to 2.

Cost

We consider the following two cost distributions in our experiments.

1. Discrete uniform distribution in [1,20] multiplying with 100.
2. The cost of activities are proportional to the duration, thus, cost of activities are calculated with multiplying the duration of activity with 100.

Note that the durations of the activities are generated with using the discrete uniform distribution in [1,10]. 80 problems are solved for each cost distribution and their results are presented in Table 12.

Table 12 – The Effect of the Cost Distribution

				Cost Distribution			
				Cost=DISC[1,20]*100		Cost = Duration *100	
N	P	M		CPU	Nodes	CPU	Nodes
35	6	1.1	Average	0.62	61739	0.62	59041
			Max	1.26	151187	2.11	180622
35	6	1.2	Average	9.04	722987	24.38	1876164
			Max	32.00	2643025	157.89	12082629
40	5	1.1	Average	40.50	3192043	7.61	654733
			Max	381.20	30398869	45.65	3933912
40	6	1.1	Average	313.65	21466377	9.28	697400
			Max	2981.40	204847218	32.20	2660936

Table 12 – The Effec of the Cost Distribution (Continued)

				Cost Distribution			
				Cost=DISC[1,20]*100		Cost = Duration *100	
N	P	M		CPU	Nodes	CPU	Nodes
40	5	1.2	Average	199.64	13807362	55.28	3580440
			Max	1714.98	120770870	267.06	16256733
45	5	1.1	Average	12.97	1000845	42.45	2633124
			Max	41.22	3276697	308.94	19393268
45	6	1.1	Average	26.90	1841934	119.11	7670742
			Max	97.42	6331793	323.39	21326518
45	6	1.2	Average	377.83	20855695	836.01	50053690
			Max	1255.62	58696652	3600 (1)	188441428

Table 12 reveals that for some combinations, the problems that have second type of costs more difficult, i.e. n=45, P=6, M=1.2; n=45, P=6, M=1.1 etc., and for some combinations the reverse holds. For example, when N=40, P=6, M=1.1, the maximum CPU seconds for the first and second types of cost distributions are 2981.40 and 32.30 seconds, respectively. As we have not seen the relative difficulty of one distribution over the other, we arbitrarily select the cost of activities to the proportional to the duration.

Duration

We try the following two duration distributions.

1. Discrete uniform distribution in [1, 10]

2. Discrete uniform distribution in [1, 20].

Table 13 reports CPU times and number of nodes statistics.

Table 13 - CPU Seconds and Nodes Analysis of Duration Distribution

				Duration			
				Duration Between [1,10]		Duration Between [1,20]	
N	P	M		CPU	Nodes	CPU	Nodes
35	6	1.1	Average	0.62	59041	5.34	460695
			Max	2.11	180622	23.68	1995346
35	6	1.2	Average	24.38	1876164	83.67	6595846
			Max	157.89	12082629	477.05	39273815
40	5	1.1	Average	7.61	654733	39.29	3169539
			Max	45.65	3933912	252.38	21573405
40	5	1.2	Average	9.28	697400	335.16	19634059
			Max	32.20	2660936	1277.90	75794148
40	6	1.1	Average	55.28	3580440	368.76	32325100
			Max	267.06	16256733	3600.00 (1)	314877398
45	5	1.1	Average	42.45	2633124	86.83	5224351
			Max	308.94	19393268	263.74	15717657
45	6	1.1	Average	119.11	7670742	298.85	18563551
			Max	323.39	21326518	1851.89	118668197
45	6	1.2	Average	836.01	50053690	2026.45	106231744
			Max	3600 (1)	188441428	3600 (5)	232629973

As can be seen from Table 13, when the durations of activities are generated from a larger interval, the variance of the problem complexity increases. Although for some of the instances with duration between [1, 20] the problem could not be solved in an hour, for some instances, the problem is solved very fast. For example, when n=40, P=6, M=1.1 the average of the 10 instances of problem is 368.76 seconds and the maximum CPU is 3600 seconds. If we exclude a single instance which could not be finished in an hour, then we can see that the average CPU of 9 instances is 9.73 seconds with a maximum of 39.39 seconds.

Bounding Mechanisms

We look for the effects of the upper bound and initial lower bounds on the branch and bound performance. For four selected combinations of problems, we compare the performance of the branch and bound algorithm that uses both initial lower

and upper bounds with those that do not the respective bounds. When no initial lower bound is used then, there is no lower bound value until the branch and bound algorithm finds the first complete solution. When there is no upper bound in the branch and bound algorithm, it implies all nodes in the tree are evaluated by their objective function values.

The lower and upper bound performances are assessed at the root node by their difference from the optimal solutions as a ratio of the optimal solution.

$$\text{Lower Bound Deviation \% (LBD)} = 100 * \frac{\text{Optimal Solution} - \text{Initial Lower Bound}}{Optimal\ Solution}$$

$$\text{Upper Bound Deviation \% (UBD)} = 100 * \frac{\text{Initial Upper Bound} - \text{Optimal Solution}}{Optimal\ Solution}$$

In Table 14, the CPU seconds and number of nodes of branch and bound algorithm, without using lower bound and without using upper bound are shown.

Table 14 – The Upper and Lower Bound CPU Performances in a BAB

N	P	M		BAB		BAB without LB		BAB without UB	
				CPU	Nodes	CPU	Nodes	CPU	Nodes
30	5	1	Average	0.02	844	0.01	974	268.17	28498128
			Max	0.06	2495	0.03	2495	925.20	97599505
35	5	1	Average	0.04	4121	0.06	4320	426.51	31522855
			Max	0.09	14879	0.19	14931	3194.53	248509551
35	5	1.1	Average	2.27	239415	3.46	293064	2267.76	195258186
			Max	18.36	1916052	22.81	1919082	3600(4)	319566329
40	6	1.2	Average	82.86	5774048	94.38	5815995	3600.00	304604987
			Max	411.97	30842679	456.12	30874349	3600(10)	351965461

Table 15 – The Deviations of the Upper and Lower Bound

N	P	M	UBD %		LBD %	
			Average	Max	Average	Max
30	5	1	2.0	3.9	0.3	0.9
35	5	1	3.5	5.2	0.4	1.6
35	5	1.1	8.4	13.7	0.7	1.6
40	6	1.2	6.7	10.0	1.7	3.4

Note from Table 14 that the effects of the upper bounds are more pronounced than the effect of the initial lower bound. The upper bounds are found so powerful, for instance, the third configuration of the problem could be solved in average 2.27 seconds with using upper and lower bounds whereas it takes approximately 1000 times longer when no upper bound is used. For the same problem combination, it takes approximately 1.5 times longer when no initial lower bound is used.

As the values are very close to the optimal objective function values, they are powerful estimators that give a conscious walk towards the optimal solution.

We are employing powerful upper bounds in a depth first strategy. This follows, the early complete solutions reached are close to the optimal solutions. Early powerful complete solutions lead to updating initial lower bound, hence reduces the need for a powerful initial lower bound. Note from our tables that, the initial lower bounds are not as effective as upper bounds. Table 15 shows that the initial lower bound is 0.7% close to the optimal solution for the third problem combination, however, the upper bound is 8.4% close to the optimal solution.

We can conclude that the effort spent to compute the bounds is justified by the reductions in the solution space. Hence, based on the observations of the upper bound and lower bound effects, we perform our main experiments using the bounds.

For our main experiment, we continue our runs with the most difficult problem combinations for profit margin, discount factor, and network complexity. Recall that the profit margin 0.20, discount factor 0.10 and network complexity 1.5, the problem complexity increases compared to profit margin 1.30, discount factor 0.05 and network complexity 2, respectively.

Based on the results of our preliminary experiments, we select the duration of activities from the discrete uniform distribution in [1, 10] and cost of activities as proportional to the duration of activities.

5.3 Main Experiments

We perform the main experiments according to the parameters given in Table 16, below. For each of combination, 10 problem instances are solved. We present the results by the deadline multiplier factors.

Table 16 – Parameters Used in the Main Experiment

Parameter	**Value**
Network size	30, 35, 40, 45, 50, 55
Number of periods	5, 6, 7
Duration	Discrete uniform distribution in [1, 10]
Cost	Duration * 100
Network Complexity	1.5
Discount Rate	10% yearly
Profit Margin	1.2
Deadline multiplier factor	1, 1.1, 1.2

In our runs, we used a computer that has 8 GB RAM and Intel Core 3.4 GHz i7 processor. Branch and bound algorithm is written in C++ language using

Microsoft Visual Studio 2012. The nonlinear mathematical model is solved in GAMS with BARON solver.

We put a CPU time limit of 1 hour to both branch and bound algorithm and mathematical model. For any problem combination, if there are more than 2 out of 10 problem instances exceed the time limit, we do not try to run the models for larger networks and larger periods. Number of problem instances that exceeds the time limit is given in parenthesis for the branch and bound algorithm in the BAB CPU seconds and for the mathematical model in the GAMS CPU seconds column.

In one hour, we could solve 450 problems by BAB and 370 problems by GAMS for our main experiment.

We first report on the lower and upper bound performances.

Table 17 – Lower Bound Performances

N	P	M=1		M=1.1		M=1.2	
		LBD %		LBD %		LBD %	
		Average	Max	Average	Max	Average	Max
30	5	0.3 (4)*	0.9	0.4 (1)	0.8	1.4	4.2
	6	0.5 (1)	1.3	0.6 (1)	0.9	0.8	1.4
	7	0.8	1.6	0.8	1.8	1.0 (1)	3.1
35	5	0.4 (4)	1.6	0.7	1.6	0.8	2.3
	6	0.7 (1)	2.6	1.3	2.4	1.9	4.3
	7	0.6 (1)	1.4	1.2	2.4	1.9	4.5
40	5	0.7	2.6	0.8	2.2	0.8	2.2
	6	0.6	1	1.8	3.5	1.7	3.4
	7	0.5	1.1	1.2	4.1	1.7	5.5
45	5	0.6	1.6	1.2	3.2	0.9	3.1
	6	0.5	1.1	1.1	2.8	1.3	3
	7	0.7	1.9	0.9	2.8	1.6	2.8

Table 17 - Lower Bound Performances (Continued)

N	P	M=1		M=1.1		M=1.2	
		LBD %		LBD %		LBD %	
		Average	Max	Average	Max	Average	Max
50	5	0.6	1.7	1.3	3.2	2.1	4.2
	6	0.7	1.7	1	2.9	-	-
	7	0.7	1.7	0.8	2	-	-
55	5	0.8	1.8	-	-	-	-
	6	1	2.5	-	-	-	-

* The number in the parenthesis gives the number of times the bound gives the optimal solution.

We observe from Table 17 that, the lower bounds work consistently well over all problem set. The average deviations are all below 1% for M=1 and 1.5 and 2% for M=1.1 and 1.2 with single exceptions of 1.8 and 2.1% respectively. We also observe the maximum deviations are below 2.6%, 4.1% and 5.5% for M=1, 1.1 and 1.2, respectively. The slightly better performances of the lower bound for lower M is that the slack values are smaller for small M, thereby there are less chances of playing with a given solution. Once you have more chances, the possibility of producing a good solution is lower. When M=1, our lower bounding procedure returns the optimal solution for 11 out of 170 solved instances. When M=1.1, two optimal solutions could be obtained, and M=1.2 only for a single instance, the optimal solution is reached. As the number of activity sizes increases the lower bound deviation slightly increases. This is due to the fact that increase in the combination which results in the increase in the problem complexity.

Table 18 – Upper Bound Performances

N	P	M=1		M=1.1		M=1.2	
		UBD %		UBD %		UBD %	
		Average	Max	Average	Max	Average	Max
30	5	2.00	3.90	3.60	5.60	5.30	7.50
	6	1.80	2.50	3.00	5.10	3.60	4.50
	7	1.30	1.90	3.00	3.90	3.30	4.20
35	5	3.50	5.20	8.40	13.70	11.20	15.20
	6	2.10	3.60	6.20	9.10	8.20	12.30
	7	1.60	2.50	4.90	7.80	5.60	7.40
40	5	2.80	4.60	7.10	11.00	8.80	14.50
	6	2.10	3.00	5.10	7.90	6.70	10.00
	7	1.50	2.80	4.00	5.80	4.50	6.90
45	5	3.60	6.10	8.50	11.40	11.10	14.20
	6	2.80	5.10	6.60	9.30	8.30	11.50
	7	2.20	2.90	5.10	7.30	6.60	8.60
50	5	3.90	4.60	9.30	13.90	12.70	17.00
	6	3.10	4.90	8.00	11.00	-	-
	7	2.00	2.90	6.20	12.20	-	-
55	5	5.60	7.20	-	-	-	-
	6	4.30	6.70	-	-	-	-

From Table 18, we observe that the performances of the upper bounds are very satisfactory. The average deviations evaluated at the root node are almost below 5%, 10% and 15% for M=1, 1.1 and 1.2, respectively. One should expect much better performances from the upper bounding procedures that would be used to evaluate the nodes that have partial information. As can be observed from Table 18, the performance of the upper bounds deteriorates as N increases and M increases. However, these increases are not in exponential rate. For example, when for M=1 and P=6, N increases from 30 to 55, the deviations increases from

1.8% to 4.3%. So, one may observe linear increases in problem size parameters for our exponentially natured problem.

As in lower bounds, the effect of the deadline values on the upper bound performances is significant. As the deadline values are higher, the slack times of the activities are higher. Higher slack times give more flexible assignments, and more choices decreases the possibility of finding a solution that is close to optimal. When N=50 and P=5, 6, and 7, the respective average deviations are 3.9, 3.1 and 2.1% for M=1, 9.3, 8 and 6.2% for M=1.1, 12.7, 9.5 and 7.5%. Note from those figures that as the number of periods increases the performance of upper bound improves. Higher P value means more chances for payments, hence the earliest periods of the activities are earlier. This reduces the difference between activity completion time and payment periods, thereby improving the performance of our upper bounds that use those ideas.

Note from all tables that lower and upper bounds work consistently well over all problem set. The maximum values are very close to the average values, that shows consistent behaviour of bounds over all problem instances.

We now report on the performance of our branch and bound algorithm. We evaluate the performance by the solution times expressed in Central Processing Unit (CPU) seconds and the number of partial solutions, i.e., nodes. We also give the average and maximum solution times of our GAMS model. Tables 19, 20, and 21 report on the average and maximum CPU times and the number of nodes for M=1, 1.1, and 1.2, respectively.

Table 19 – The Performances of the BAB and GAMS model, M=1

N	P	BAB CPU seconds		GAMS CPU seconds		Nodes	
		Average	Max	Average	Max	Average	Max
30	5	0.02	0.06	2.03	7.26	844	2495
	6	0.03	0.07	4.21	18.8	3066	10107
	7	0.02	0.06	4.23	11.91	1702	5049
35	5	0.04	0.09	2.24	5.29	4121	14879
	6	0.03	0.13	2.1	5.25	3370	18067
	7	0.17	1.39	3.54	17.41	18760	165085
40	5	0.24	1.44	11.39	38.78	24141	154159
	6	0.26	1.08	5.88	13.34	28495	116463
	7	0.22	1.19	11.2	44.51	21869	108400
45	5	2.69	25.1	37.38	241.21	208874	1965421
	6	0.8	3.29	234.92	1667.84	66955	303130
	7	0.92	5.43	100.84	585.59	75416	473736
50	5	3.18	11.72	287.02	1584.23	194543	832767
	6	11.96	34.41	628.52	3600 (1)	665025	1998490
	7	11.28	47.03	722.76	3600 (1)	520479	2388009
55	5	258.61	1416.44	1489.91	3600 (3)	12931617	82941760
	6	1293.75	3600(3)	-	-	63119309	179327664

Table 20 - The Performances of the BAB and GAMS model, M=1.1

N	P	BAB CPU seconds		GAMS CPU seconds		Nodes	
		Average	Max	Average	Max	Average	Max
30	5	0.04	0.11	11.9	60.03	4953	13855
	6	0.1	0.26	18.05	72.51	11464	33445
	7	0.33	1.61	14.15	54.62	42107	216955

Table 20 - The Performances of the BAB and GAMS model, M=1.1 (Continued)

N	P	BAB CPU seconds		GAMS CPU seconds		Nodes	
		Average	Max	Average	Max	Average	Max
35	5	2.27	18.36	68.42	237.86	239415	1916052
	6	0.62	2.11	51.78	236.51	59041	180622
	7	9.09	48.51	128.79	543.28	915984	4878913
40	5	7.61	45.65	86.49	260.74	654733	3933912
	6	9.28	32.2	252.12	1321.71	697400	2660936
	7	11.26	75.05	229.81	809.19	912521	6348481
45	5	42.45	308.94	999.13	3600(2)	2633124	19393268
	6	119.11	323.39	-	-	7670742	21326518
	7	36	151.42	-	-	2277277	10110373
50	5	487.13	2496.95	-	-	19454785	93547006
	6	819.99	3600(1)	-	-	39658950	174604990
	7	1001.02	3600(2)	-	-	46591672	163661586

Table 21 - The Performances of the BAB and GAMS model, M=1.2

N	P	BAB CPU seconds		GAMS CPU seconds		Nodes	
		Average	Max	Average	Max	Average	Max
30	5	0.46	1.66	10.9	25.91	37582	138031
	6	0.39	1.03	15.14	44.71	52674	138933
	7	0.66	2.37	11.48	21.99	69205	245846
35	5	9.64	40.6	279.67	843.18	708085	2965672
	6	24.38	157.89	562.51	3102.25	1876164	12082629
	7	11.27	47.67	171.59	794.33	662745	2892718
40	5	55.28	267.06	546.59	1866	3580440	16256733
	6	82.86	411.97	807.85	3600.00 (1)	5774048	30842679
	7	122.79	788.1	566.35	2397.19	8331816	49267279

Table 21 - The Performances of the BAB and GAMS model, M=1.2 (Continued)

N	P	BAB CPU seconds		GAMS CPU seconds		Nodes	
		Average	Max	Average	Max	Average	Max
45	5	189.11	1004.6	2797.36	3600(6)	11290379	62954089
	6	836.01	3600(1)	-	-	50053690	188441428
	7	1033.79	3600(1)	-	-	61144541	254838453
50	5	1834.04	3600(4)	-	-	94182551	247572280

From Tables 19, 20, and 21, we observe the significant effect of the number of activities on the complexity of the solutions. When deadline multiplier is 1, the BAB could solve the instances up to 55 activities, when deadline multiplier is 1.1 and 1.2, the instances with up to 50 activities could be solved. The maximum number of periods that could be solved by deadline values of 1.1 for 50 is 7, whereas a maximum of 5 periods could be solved for a deadline multiplier value of 1.2. Those results altogether lead us to conclude that the number of activities, number of periods and deadline values are affecting the performance. The higher values of all parameters, the higher is the complexity of the solutions.

We observe that among those three parameters N is the most significant one that affects the performance. The CPU times by BAB are below 1 second when N=30 for all values P and M. For M=1, when N becomes 55 for P=5 and 6, the average CPU times are 258.61 and 1293.75 seconds, respectively. 3 out of 10 problem could not be solved when P=6. For M=1.1 and 1.2 we observe that, 55 activity problems, on the other hand, could be solved in 1834 seconds on average for P=5, leaving 4 unsolved problems in 1 hour. When P=6, no 50 activity instance could be solved. For M=1.1, the average CPU times are 487.13, 819.99 and 1001.02 seconds, for P=5, 6 and 7, respectively. When P=6 and 7, 1 and 2 problems respectively, could not be solved in 1 hour.

Note that as N increases, the size of the problem, hence the size of the branch and bound tree increases exponentially. Our branch and bound algorithm has dispelled this exponential effect, at some extent, as it uses very powerful upper bounds. As our upper bounds are good estimators of the optimal objective function value, they lead to good choices between nodes, thereby decreasing the size of the tree. Recall that our lower and bound performances are not much sensitive to the increases in the parameter values.

We also observe the significant effect of deadline multipliers on the BAB performance. As M increases, for each interval the number of choices increases and this increase the effort spent to make evaluations. For smaller M, many assignments lead to infeasible solutions, thereby the size of the search is smaller. Moreover, our upper bounds perform better for smaller M, hence their effects in a BAB would be higher. Note from Tables 19, 20, and 21 that we obtain smallest CPU times at the lowest M value. When N=45 and P=7, the average CPU times are 258.61 487.13 and 1834.04, for M=1, 1.1, 1.2, respectively. When N=50, P becomes 7, the average CPU times for M=1, and 1.1 are 11.28 and 1001.02 respectively. No instance could be solved in 1 hour for M=1.2.

P also affects the BAB performance, and the effect becomes more significant as the problem complexity due to N or M is high. As P increases, the number of times that BAB gives decisions increases the CPU times. Note that when N=55 and M=1, the average CPU times increase from 258.61 to 1293.75 as P increases from 5 to 6. When N=50 and M=1.1, the average CPU times increase from 487.13 to 1001.02 as P increases from 5 to 7. There are some exceptions, where an increase in P, reduces the CPU time. For example, according to Table 19, for N=45, and M=1, the average CPU time with 5 period is 2.69 and this is higher average CPU time with 6 periods (0.80 sec).

This unexpectedly big values for 5 periods, is due to a single dominating instances with CPU time of 25.1. Once we exclude this instance, the average over 9 instances reduces 0.2.

CHAPTER 6

CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

In this study we consider a project payment model with discounted cash flows. We assume that the client payment times and project deadline are defined in the project contract. The activities are characterized by their processing times and costs that are incurred at their completions. Our problem is to find the client payment amounts and activity completion times so as to maximize the net present value of the client payments and activity costs. We show that the problem is strongly NP-hard.

We formulate the problem as a mixed integer nonlinear programming model and solve the instances up to 55 activities, 5 periods when deadline is tight. For loose deadline case, the instances up to 45 activities, 5 periods could be solved by the model.

For moderate to large sized problem instances, we propose a branch and bound algorithm that employs efficient lower and upper bounding mechanisms. We find that algorithm is able to solve the problems up to 55 activities, 6 periods when deadline is tight. For loose deadline case, the instances up to 50 activities, 5 periods could be solved by the algorithm.

The results of our extensive computational experiments have revealed that the most significant parameter is the number of the activities. As number of the activities increase the solution times increase considerably. We also find that the

number of the periods and deadline are dominant parameters that affect the solution times. The higher solution times are observed when the number of the periods is higher and the deadline is looser.

To the best of our knowledge our study is the first attempt to solve the progress payment problem where the payment quantities are to be decided at defined time points. We hope the results of our study could help for the practitioners who want to manage their cash flows, negotiating for the milestones and critical control points, determining the amounts and timings of the progress payments. Our models and algorithms can be easily modified to handle the practical cases where the costs are charged once the activities start and/or progress payments are received at the beginning of the periods.

This study has considered a project payment model from contractor's view. Extending the results so as to emphasize the client's view as well as joint views of contractor and client might be considered in future work. Our models can also be extended for bonus payment and penalty cost structures.

In this study, we assume that the profit margin and the discount rate do not change as time progresses. The future research may relax those assumptions and extend our procedures to handle the dynamic values for the profit margin and discount rate.

Future research may also consider the design of more powerful optimization algorithms that use Bender's Decomposition ideas as suggested in Grinold (1972). Moreover heuristic approaches that provide high quality schedules, in reasonable times might be a designed. Our lower bounding procedure can be used as a stepping-stone for developing powerful heuristic approaches.

REFERENCES

Chen, W. N., Zhang, J., Chung, H. S. H., Huang, R. Z., and Liu O. (2010), "*Optimizing Discounted Cash Flows in Project Scheduling – An Ant Colony Optimization Approach",* IEEE Transactions on Systems, Man, and Cybernetics – Part C. Applications and Reviews, 40, 64-77.

Dayanand, N., and Padman, R. (1997), "*On Modelling Payments in Projects",* The Journal of the Operational Research Society, 48, 906-918.

De Reyck, B., and Herroelen, W. (1996), "*An Optimal Procedure for the Unconstrained Max-npv Project Scheduling Problem with Generalized Precedence Relations",* Department of Applied Economics, Katholieke Universiteit Leuven, Research Report, 9642.

De Reyck, B., and Herroelen, W. (1998), "*An Optimal Procedure for the Resource-Constrained Project Scheduling Problem with Discounted Cash Flows and Generalized Precedence Relations",* Computers and Operations Research, 25, 1-17.

De Reyck, B., and Leus, R. (2008), "*R&D Project Scheduling when Activities may Fail",* IIE Transactions, 40, 367-384

Erenguc, S. S., Tufekci, S., and Zappe, C. J., (1993), "*Solving Time/Cost Trade-off Problems with Discounted Cash Flows Using Generalized Benders Decomposition*", Naval Research Logistics, 40, 25-50.

Grinold, R. G., (1972), "*The Payment Scheduling Problem*", Naval Research Logistics Quarterly, 19, 123-136.

He, Z., and Xu, Y. (2008), "*Multi-Mode Project Payment Scheduling Problems with Bonus-Penalty Structure",* European Journal of Operational Research, 189, 1191-1207.

He, Z., Wang, N., Jia, T., and Xu, Y. (2009), "*Simulated Annealing and Tabu Search for Multi-Mode Project Payment Scheduling",* European Journal of Operational Research, 198, 688-696.

Herroelen, W. S., Van Dommelen, P., and Demeulemeester E. L. (1997), "*Project Network Models with Discounted Cash Flows a Guided Tour through Recent Developments",* European Journal of Operational Research, 100, 97-121.

Icmeli, O., and Erenguc, S.S. (1996), *"A Branch and Bound Procedure for the Resource Constrained Project Scheduling Problem with Discounted Cash Flows"*, Management Science, 42, 1395-1408.

Kazaz, B., and Sepil, C. (1996), "*Project Scheduling with Discounted Cash Flows and Progress Payments",* The Journal of the Operational Research Society, 47, 1262-1272.

Kolisch, R., and Sprecher, A. (1996), "*PSPLIB – A Project Scheduling Problem Library",* European Journal of Operational Research, 96, 205-216.

Mika, M., Waligora, G., and Weglarz, J. (2005), "*Simulated Annealing and Tabu Search for Multi-Mode Resource-Constrained Project Scheduling with Positive Discounted Cash Flows and Different Payment Models",* European Journal of Operational Research, 164, 639-668.

Najafi, A. A., and Niaki, S. T. A. (2005), "*Resource Investment Problem with Discounted Cash Flows",* International Journal of Engineering, 18, 100-101.

Najafi, A. A., and Azimi, F. (2009), "*A Priority Rule-Based Heuristic for Resource Investment Project Scheduling Problem with Discounted Cash Flows and Tardiness Penalties",* Hindawi Publishing Corporation, Mathematical Problems in Engineering, 2009, ID 106425.

Özdamar, L., and Dündar, H. (1997), "*A Flexible Heuristic For a Multi-Mode Capital Constrained Project Scheduling Problem with Probabilistic Cash Inflows",* Computers and Operational Research, 24, 1187-1200.

Ritwik, A., and Paul, G. (2013), "*A Heuristic Algorithm for Resource Constrained Project Scheduling Problem with Discounted Cash Flows",* International Journal of Innovative Technology and Exploring Engineering, 3, 99-102.

Russell, A. H. (1970), "*Cash Flows in Networks",* Management Science, 16, 357-373.

Seifi, M., and Tavakkoli-Moghaddam, R. (2008), "*A New Bi-Objective Model for a Multi-Mode Resource-Constrained Project Scheduling Problem with Discounted Cash Flows and Four Payment Models",* IJE Transactions A: Basics, 21, 347-360.

Sepil, C., and Ortaç, N. (1997), "*Performance of the Heuristic Procedures for Constrained Projects with Progress Payments",* Operational Research Society, 48, 1123-1130.

Smith-Daniels, D.E., Padman, R., and Smith-Daniels, V.L. (1996), "*Heuristic Scheduling of Capital Constrained Project",* Journal of Operations Management, 14, 241-254.

Ulusoy, G., and Cebelli, S. (2000), "*An Equitable Approach to the Payment Scheduling Problem in Project Management",* European Journal of Operational Research, 127, 262-278.

Ulusoy, G., Sivrikaya-Şerifoğlu, and F., Şahin, Ş. (2001), "*Four Payment Models for the Multi-Mode Resource Constrained Project Scheduling Problem with Discounted Cash Flows",* The Annals of Operations Research, 102, 237-261.

Vanhoucke, M., Demeulemeester, E., Herroelen, W. (2003), "*Progress Payments in Project Scheduling Problems"*, European Journal of Operational Research, 148, 604-620.

APPENDIX A

BRANCH AND BOUND CALCULATIONS

Table A1 – ESS and LSS of Nodes 0, 8, 16

Node	Act	ES	EF	LS	LF	PES	PLS
0	1	0	0	0	0	1	1
	2	0	3	7	10	1	1
	3	0	2	13	15	1	2
	4	3	8	10	15	1	2
	5	3	9	17	23	1	3
	6	8	16	15	23	2	3
	7	16	20	23	27	2	3
	8	27	30	27	30	3	3
	9	16	20	26	30	2	3
	10	30	30	30	30	3	3
8	1	0	0	0	0	1	1
	2	0	3	4	7	1	1
	3	0	2	10	12	1	2
	4	3	8	7	12	1	2
	5	3	9	14	20	1	2
	6	8	16	12	20	2	2
	7	23	27	23	27	3	3
	8	27	30	27	30	3	3
	9	26	30	26	30	3	3
	10	30	30	30	30	3	3

Table A1 – ESS and LSS of Nodes 0, 8, 16 (Continued)

Node	Act	ES	EF	LS	LF	PES	PLS
16	1	0	0	0	0	1	1
	2	2	5	2	5	1	1
	3	8	10	8	10	1	1
	4	5	10	5	10	1	1
	5	14	20	14	20	2	2
	6	12	20	12	20	2	2
	7	23	27	23	27	3	3
	8	27	30	27	30	3	3
	9	26	30	26	30	3	3
	10	30	30	30	30	3	3