DISCRETE TIME/COST TRADE-OFF PROBLEM IN PROJECT SCHEDULING

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

BY

AHMET BAYKAL HAFIZOĞLU

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

JUNE 2007

Approval of the Graduate School of Natural and Applied Sciences

_____

Prof. Dr. Canan Özgen
Director

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

_____

Prof. Dr. Çağlar Güven
Head of Department

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

_____

Prof. Dr. Meral Azizoğlu
Supervisor

Examining Committee Members

Prof. Dr. Ömer Kırca (METU, IE) _____

Prof. Dr. Meral Azizoğlu (METU, IE) _____

Asst. Prof. Dr. F. Can Çetinkaya (Çankaya Univ., IE) _____

Assoc. Prof. Dr. Canan Sepil (METU, IE) _____

Asst. Prof. Dr. Ayten Türkcan (METU, IE) _____

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: Ahmet Baykal Hafızoğlu

Signature :

ABSTRACT

DISCRETE TIME/COST TRADE-OFF PROBLEM IN PROJECT SCHEDULING

HAFIZOĞLU, Ahmet Baykal
M.S., Department of Industrial Engineering
Supervisor: Prof. Dr. Meral AZİZOĞLU

June 2007, 86 pages

In project scheduling, the activity durations can often be reduced by dedicating additional resources. Time/Cost Trade-off Problem considers the compromise between the total cost and project duration. The discrete version of the problem assumes a number of time/cost pairs, so called modes, and selects a mode for each activity.

In this thesis we consider the Discrete Time/Cost Trade-off Problem. We first study the Deadline Problem, i.e., the problem of minimizing total cost subject to a deadline on project duration. To solve the Deadline Problem, we propose several optimization and approximation algorithms that are based on optimal Linear Programming Relaxation solutions. We then analyze the problem of generating all efficient solutions, and propose an approach that uses the successive solutions of the Deadline Problem.

Our computational results on large-sized problem instances have revealed the satisfactory behavior of our algorithms.

Keywords: Project Scheduling, Time/Cost Trade-off, Branch and Bound

ÖZ

PROJE ÇİZELGELEMESİNDE KESİKLİ ZAMAN/MALİYET ÖDÜNLEŞİM PROBLEMİ

HAFIZOĞLU, Ahmet Baykal
Yüksek Lisans, Endüstri Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Meral AZİZOĞLU

Haziran 2007, 86 sayfa

Proje çizelgelemesinde aktivite süreleri ek kaynaklar tahsis edilerek azaltılabilir. Zaman/Maliyet Ödünleşim Problemi toplam maliyet ve proje süresi arasındaki uzlaşmayı ele alır. Problemin kesikli versiyonu mod diye tabir edilen, belirli sayıda zaman/maliyet çiftleri varsayar ve her aktivite için bir mod seçer.

Bu tezde Kesikli Zaman/Maliyet Ödünleşim Problemini ele aldık. Öncelikle Zaman Sınırı Problemi, başka bir deyişle, proje bitirme süresi sınırına bağlı toplam maliyet enazlama problemi üzerinde çalıştık. Zaman Sınırı Problemini çözmek için, optimal doğrusal programlama gevşetmesine dayanan eniyileme ve yaklaşıklama algoritmaları önerdik. Sonra Tüm Verimli Çözümleri Üretme Problemini inceledik, ve Zaman Sınırı Probleminin ardışık çözümlerini kullanan bir yaklaşım önerdik.

Büyük ölçekli problem örneklerindeki sonuçlarımız algoritmalarımızın memnuniyet verici tutumunu göstermektedir.

Anahtar Kelimeler: Proje Çizelgelemesi, Zaman/Maliyet Ödünleşimi, Dal-Sınır Yöntemi

To my family

ACKNOWLEDGEMENTS

I am deeply grateful my thesis supervisor Prof. Meral Azizoğlu for her efforts, guidance and support throughout the study. She not only guided me perfectly throughout the study, but also helped me to develop a bright career as an ideal figure of an academic.

I would like to thank jury members for their valuable contributions on the thesis.

I would like to express my deepest appreciation to my family members: Harzemşah Hafızoğlu, Nuran Hafızoğlu and especially İdil Hafızoğlu for their moral support I received throughout my thesis. Without their love, such a work would not exist.

I would like to thank my dear friends Melih Özlen for his invaluable support and guidance; Banu Lokman for her help, moral support and cheerful company; Tülin İnkaya and Bora Kat for their company even in the hardest parts of this study; Pelin Bayındır for her insight and wisdom; Mustafa Gökçe Baydoğan for his support and invaluable friendship. I would also like to thank to colleagues İbrahim Karahan, Nihan Görmez, Oğuz Solyalı and Selin Bilgin for providing me a working environment I will always miss.

I would also like to thank to all occupants of 63/9, namely, Ertuğrul Berk Gürtekin, Musa Arda and Mehmet Akten and friends Erhan Akbaş, Rafet Çakır, Eda Çuvaloğlu and Gülçin Haktanır for their friendship and encouragement in the hardest pieces of this study.

And finally, thanks to our friend in the radio, rain clouds, “green”day and foxes…

TABLE OF CONTENTS

ABSTRACT v
ÖZ……..... vi
ACKNOWLEDGEMENTS viii
TABLE OF CONTENTS ix
LIST OF TABLES xi
LIST OF FIGURES xiii
CHAPTER
1 INTRODUCTION 1
2 PROBLEM DEFINITION AND RELATED LITERATURE 4
2.1 Project Scheduling Problems 4
2.2 Formulations for Multi-Mode Problems 11
2.3 Literature Search on Multi-Mode Problems 15
3 THE DEADLINE PROBLEM 20
3.1 Problem Size Reduction 20
3.2 Lower Bounds 27
3.2.1 Naive Bound, LB_1 27
3.2.2 LP Relaxation Based Bound, LB_2 28
3.3 Branch and Bound Algorithm 31
3.4 Upper Bounds 34
4 THE TIME/COST CURVE PROBLEM 40
5 COMPUTATIONAL RESULTS 50
5.1 Data Generation 50
5.2 Performance Measures 52
5.3 Preliminary Experiments 53
5.4 Main Experiment 57
5.4.1 Deadline Problem 58
5.4.2 Time/Cost Curve Problem 68
6 CONCLUSIONS 71

REFERENCES 73
APPENDICES
A. NUMERICAL EXAMPLE ON HEURISTICS 76
B. PRELIMINARY RUN RESULTS OF HEURISTIC BRANCH AND BOUND ALGORITHM 2 83
C. CPLEX AND BRANCH AND BOUND ALGORITHM CPU TIMES 84
D. DETAILED RESULTS OF TIME/COST CURVE PROBLEM 86

LIST OF TABLES

TABLES

Table 2.1: Sample Project Network Precedence Relations 5
Table 2.2: Modes of an Example Instance 7
Table 2.3: CPM Calculations of an Example 11
Table 3.1: CPM Calculations Summary with Shortest Activity Durations (Iteration 1 Results with t =11) 23
Table 3.2: CPM Calculations Summary with Longest Activity Durations (Iteration 1 Results with t =11) 24
Table 3.3: CPM Calculations Summary with Shortest Activity Durations (Iteration 2 Results with t =11) 25
Table 3.4: CPM Calculations Summary with Shortest Activity Durations (Iteration 1 Results with t =18) 25
Table 3.5: CPM Calculations Summary with Longest Activity Durations (Iteration 1 Results with t =18) 26
Table 3.6: CPM Calculations Summary with Shortest Activity Durations (Iteration 2 Results with t =18) 27
Table 3.7: Modes and Precedence Relations for Example Problem 32
Table 3.8: The Optimal LP Solution, i.e., LB_2 33
Table 4.1: All Feasible Solutions of Sample Network 46
Table 5.1: Time/Cost Curve Problem Parameters 51
Table 5.2: Deadline Problem Parameters 52
Table 5.3: Branch and Bound Algorithm Versions 53
Table 5.4: Average CPU Times for Mode Elimination Algorithms 54
Table 5.5: Average CPU Time (seconds) for Branching and Selection Strategies 55
Table 5.6: Average Number of Nodes for Branching and Selection Strategies 55
Table 5.7: Average CPU Times of the Large-Sized Instance 55
Table 5.8: Average CPU Times of BAB Using Different Upper Bounds 56

Table 5.9: Average Number of Nodes of BAB Using Different Upper Bounds.....57
Table 5.10: Lower Bound Deviations.....59
Table 5.11: Branch and Bound Algorithm Results for Deadline Problem 61
Table 5.12: Construction Heuristic Results 63
Table 5.13: Improvement Heuristic 1 Results.....64
Table 5.14: Improvement Heuristic 2 Results.....65
Table 5.15: Heuristic Branch and Bound Algorithm 1 Results.....66
Table 5.16: Comparison of Heuristics 69
Table 5.17: Time/Cost Curve Problem CPU Times (in seconds) 70
Table A.1: Data for the Example Instance 76
Table A.2: Fractional Variables of the LP Relaxation.....77
Table A.3: Initial Improvements in Improvement Heuristic 1 78
Table A.4: Summary of the Execution of the Improvement Heuristic 1 79
Table A.5: Improvements of Activity 8 at Improvement Heuristic 2.....81
Table B.1: Average CPU Times and Deviations of 4 Small-Sized Instances.....76
Table B.2: Average CPU Times and Deviations of the Large-Sized Instance 76
Table C.1: CPLEX and Branch and Bound Algorithm CPU Times (in seconds) for CI=13 Instances 77
Table C.2: CPLEX and Branch and Bound Algorithm CPU Times (in seconds) for CI=14 Instances 77
Table D.1: Time/Cost Curve Results.....86

LIST OF FIGURES

FIGURES

Figure 1.1: Time/Cost Trade-off Problem Categories 2
Figure 2.1: Activity on Node Representation of Sample Network 5
Figure 2.2: Activity on Arc Representation of Sample Network 6
Figure 2.3: CPM Example Network 9
Figure 2.4: Critical Path 11
Figure 3.1: AoA Representation of the Sample Project Network 23
Figure 3.2: Sample Solution Space 28
Figure 3.3: Sample Solution Space with Mode Elimination 29
Figure 3.4: Branching Scheme 31
Figure 3.5: Alternate Branching Scheme 34
Figure 3.6: Alternate Branching Scheme 39
Figure 3.7: Heuristic Branch and Bound Algorithm Scheme 39
Figure 4.1: Particular Solutions to Time/Cost Curve Problem 40
Figure 4.2: Solution Space and Efficient Solutions Space 43
Figure 4.3: AoA Representation of the Sample Project Network 45
Figure 4.4: Solution Space of the Sample Project Network 46

CHAPTER 1

INTRODUCTION

In project scheduling, a set of activities with specified precedence relations has to be performed so as to create a service or product. An activity of the project can be performed in several different ways with different times and/or costs. Usually, it may be possible to accelerate the process by reducing the activity durations. The activity durations can be reduced by dedicating extra resources. These resources can be human, tool, machine, alternative processing options, like the subcontracting. An increase in resource consumption naturally increases the cost and decreases the time. According to the resource consumption levels, the alternatives can be defined. Each alternative defines its own cost and time combination and is referred to as mode in project scheduling terminology.

The number of modes for each activity is usually limited and the decision is to select an activity mode that leads promising project outcomes. As increasing the duration of an activity, reduces its cost, such problems are referred to as Time/Cost Trade-off Problems.

Time/Cost Trade-off Problems have been studied by many researchers and for many years. The first studies date back to early sixties. There are several categories of the problem. Figure 1.1 below demonstrates the classification scheme of Time/Cost Trade-off Problems, according to the nature of time/cost functions.

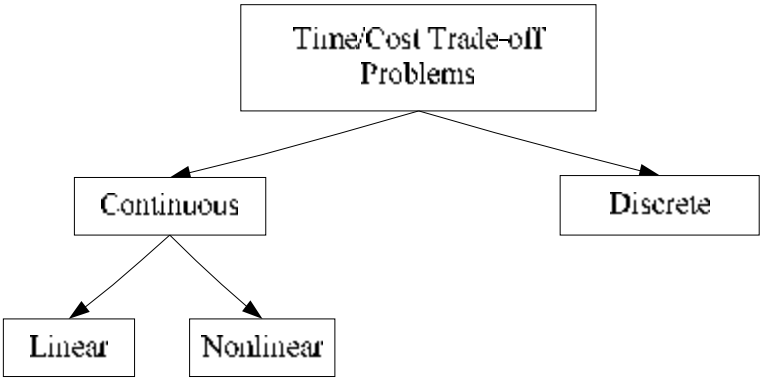

Figure 1.1: Time/Cost Trade-off Problem Categories

The classification used is based on the relation of time and cost figures. Problems with continuous time cost relations are studied in two subsections: Linear time/cost relations and nonlinear time/cost relations.

The objective function and the constraints of the problem give rise to three different Discrete Time/Cost Trade-off Problem categories:

- Deadline Problem
- Budget Problem
- Time/Cost Curve Problem

The Deadline Problem assumes an upper limit on the duration of the project, i.e., the maximum completion time over all activities. The aim is to minimize the total cost. The problem is shown to be NP-hard in the strong sense by Dunne *et al.* (1997).

The Budget Problem assumes an upper limit on the available amount of resources allocated. The objective is to minimize the project duration. A special case of the problem where all activities have a single mode is easily solvable by the Critical Path Method. The Critical Path Method simply finds the minimum project duration with given activity durations. The problem is shown to be NP-hard in the strong sense by Dunne *et al.* (1997).

The Time/Cost Curve Problem generates all efficient solutions relative to the total cost, and project duration objectives. A non-dominated, i.e., an efficient, set is defined using the successive solutions of the Deadline or Budget Problems. The Time/Cost Curve Problem is strongly NP-hard, as the Deadline Problem and Budget Problem is strongly NP-hard.

In this study, we first consider the Deadline Problem for the discrete time/cost alternatives. We propose a Branch and Bound Algorithm and several heuristic procedures. All procedures are based on the Linear Programming Relaxations of the problem. We define the properties of the Linear Programming Relaxation and use them in designing our algorithms. We then consider a Discrete Time/Cost Curve Problem. Our approach uses the successive solutions of the Deadline Problem. We propose optimization and approximation algorithms to solve the curve problem.

In the literature, the optimization procedures designed for the Deadline and Time/Cost Curve Problems could solve only moderate-sized instances up to 50 activities and 4 modes. Our algorithms, on the other hand, solve large-sized instances with up to 150 activities and about 10 modes in reasonable times.

The rest of the thesis is organized as follows. In Section 2, we define the problems and discuss the related literature. Section 3 presents our work on the Deadline Problem. The properties of the optimal solution, our Branch and Bound Algorithm and heuristic algorithms are discussed. In section 4, we analyze the Time/Cost Curve Problem. Our approach that uses successive solutions of the Deadline Problem is presented. Section 5 reports our computational experiment and its results. We conclude in Section 6 by pointing our main findings and suggestions for future research.

CHAPTER 2

PROBLEM DEFINITION AND LITERATURE REVIEW

In this chapter, we first define single and multi-mode project scheduling problems. We then give the mathematical representations for the different versions of the multi-mode problems. Finally, we review the previous studies of the multi-mode problems.

2.1 Project Scheduling Problems

Project is the process of creating a service or product by the contribution of jobs and resources within an organizational order. Project management is the discipline of planning and organization of jobs and resources in order to satisfy project constraints. In many projects resources can be represented by time and money. Many projects may not allow excess use of resources. Therefore, project time and project budget may be limiting factors. Some jobs may require some other jobs to be completed before they begin. These relations between jobs are called precedence relations which constitute another body of project constraints. In project scheduling terminology jobs are referred as activities. The relations of activities are best represented by project networks.

Project Networks

In the literature there are two different representations used for project networks: Activity on Node (AoN) Representation and Activity on Arc (AoA) Representation. These representations are separated in terms of activity and precedence relations. To define the precedence relations, we use the predecessor and successor activities. If start of activity i requires, activity j to be completed we say activity j is the predecessor of activity i, and activity i is the successor of activity j. If activity i can start immediately after activity j, then activity j is the immediate predecessor of activity i, and activity i is the immediate successor of activity j.

Activity on Node (AoN) Representation

In AoN representation, the activities are shown by nodes. Arcs represent the immediate precedence relations. The direction of an arc shows the precedence direction. Assume we have an 8 activity project having precedence relations as shown in Table 2.1. AoN representation of the project network is given in Figure 2.1.

Table 2.1: Sample Project Network Precedence Relations

Activity	Immediate Predecessor	Immediate Successor
1	-	2,3
2	1	4,5
3	1	7
4	2	6
5	2	8
6	4	8
7	3	8
8	5,6,7	-

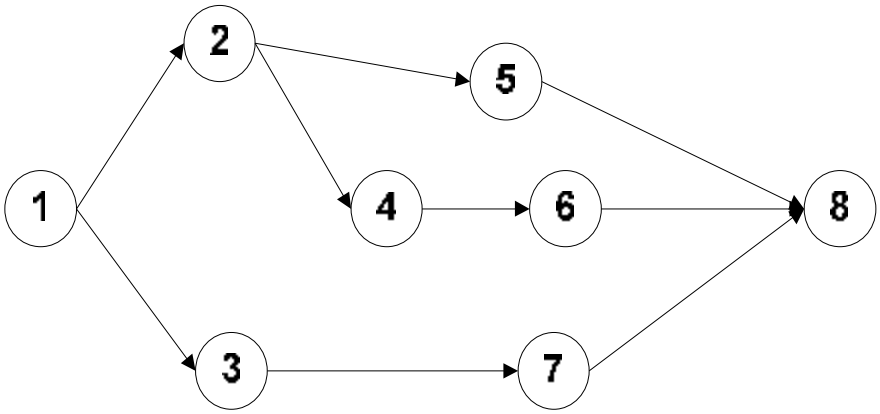

Figure 2.1: Activity on Node Representation of Sample Network

Note that activities 1 and 2 are predecessors of activity 4, and activities 6 and 8 are successors of activity 4.

Activity on Arc (AoA) Representation

According to AoA representation scheme the activities are shown by arcs. All activities have a source node and a sink node. The source node of an activity represents an event of starting that activity whereas its sink node represents an event of completing that activity. The arcs representing the immediate successor activities of an activity should start from the sink node of the activity. Therefore, the corresponding node becomes the source node of the immediate successor nodes. In project networks, parallel arcs are not allowed, hence in some cases it is required to use dummy activities. AoA representation is more widely used to demonstrate project networks. AoA representation for the sample network, whose data are given in Table 2.1, is shown in Figure 2.2.

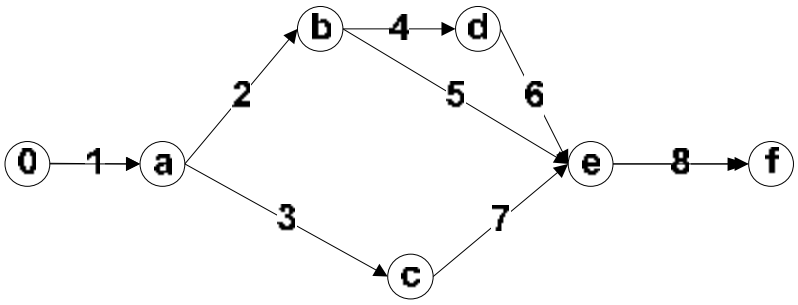

Figure 2.2: Activity on Arc Representation of Sample Network

In Figure 2.2, node 0 is an event for the start of activity 1, therefore, the start of the project. Node *b* represents the completion of activity 2 and start of activities 4 and 5. Node *f* represents the completion of activity 8, therefore, the completion of the entire project. The completion time of the last activity 8 defines the project completion time. The project completion time is also referred to as project duration.

In project scheduling problems each activity has a duration, and a cost. Occasionally, there may be alternative time/cost values for any activity. In such cases, the durations of the activities are inversely proportional with their cost values. For instance, in a tunnel project, drilling operation can be completed with a single drilling machine in a long time. To decrease the operation duration, additional drilling machines can be assigned to the operation. In such a case, the operation duration decreases; however, money paid to drilling machines increases. As more

drilling machines are assigned to the operation, duration continuously decreases and total cost increases. The time/cost pair for each alternative situation is called mode. Mode is simply the word used instead of "alternative" in project scheduling terminology. Table 2.2 illustrates the data for a 5-mode problem.

Table 2.2: Modes of an Example Instance

Mode	Total Cost	Time	# of Machines Used
1	$100000	12 months	1
2	$200000	8 months	2
3	$300000	5 months	3
4	$400000	3 months	4
5	$500000	2 months	5

As can be observed from the table, the time decreases, with cost increases due to the additional resources, i.e., machines used.

When there is more than one mode for an activity, a decision has to be made about which mode to choose. The associated problem is choosing right modes to minimize project duration or total cost without violating the project constraints. These kinds of problems are referred as ***Multi-mode*** problems.

When there is single mode for each activity the problem is called ***Single-mode*** problem.

Single-Mode Problems

The problem is to determine the project duration. To find minimum project duration Critical Path Method (CPM) is used. Since there is only one mode for each activity, total project cost is simply the summation of costs of all activities.

For single-mode problems the parameters that define the problem are precedence relations and duration of activities. t_i is the duration of activity i. t_i is also referred as activity times. P_i (S_i) is the set of immediate predecessors (successors) of activity i. The aim is to find the minimum project duration T.

For a given project, the following information can be derived:

ES_i: Earliest possible start time of activity i

LC_i: Latest possible completion time of activity i

Slack time of activity i: Maximum possible duration of activity *i* without increasing the project duration. It is simply the difference between latest possible completion and earliest possible start time.

Critical Activity: The activity whose slack time is equal to its duration. In other words, any activity is called critical if an increase in its duration directly affects the project duration. All activities which are not critical are called ***non-critical activities***. The set of critical activities are denoted as ***Crit***.

To summarize, we have the following relations:

$$Slack_i = LC_i - ES_i \qquad i = 1, 2, ..., N$$

$$Crit = \{i = 1, 2, ..., N \mid Slack_i = t_i\}$$

Critical Path: A path from source to sink consisting of all critical activities.

The CPM algorithm works as follows:

In initialization step, the earliest start times of the activities without predecessors are set to 0. Then the earliest start times of other activities are calculated. For each activity, the earliest start times of all immediate predecessors are known. The earliest start time of activity is the maximum of the earliest completion times of its predecessors. After all earliest start times are computed, the earliest project duration is found, which is the maximum of all earliest completion times. The latest completion times are then computed, starting from the activities having no successors. The latest completion time of an activity is the minimum of the latest start times of all its immediate successors. After all earliest start and latest completion times are found, the slack times are computed and the critical activities are detected. Below is the stepwise description of the CPM.

Initialization:

$ES_i = 0 \qquad i : P_i = \varnothing$

Main Body:

Repeat

$$ES_i = \underset{j \in P_i}{Max}\{ES_j + t_j\} \qquad i : \forall j \in P_i \; ES_j \text{ is calculated}$$

Until ES_i for $i = 1, 2, ..., N$ are calculated

$$T = \underset{i}{Max}\{ES_i + t_i\}$$

$LC_i = T \qquad i : S_i = \varnothing$

Repeat

$$LC_i = \underset{j \in S_i}{Min}\{LC_j - t_j\} \qquad i : \forall j \in S_i \; LC_j \text{ is calculated}$$

Until LC_i for $i = 1, 2, ..., N$ are calculated

Finalization:

$Slack_i = LC_i - ES_i \qquad i = 1, 2, ..., N$

$Crit = \{i = 1, 2, ..., N \mid Slack_i = t_i\}$

We illustrate the CPM method on the project instance whose data were given in Table 2.1. The AoA representation is given in Figure 2.3 below. The numbers in the parentheses on the arcs denote the activity durations.

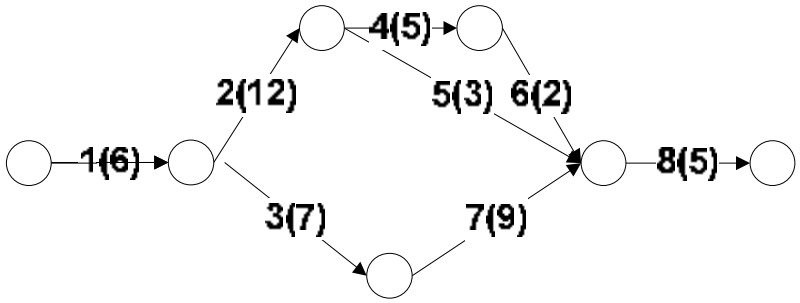

Figure 2.3: CPM Example Network

We now give the stepwise implementation of the method.

Initialization:

Activity 1 has no immediate predecessor. Therefore $ES_1 = 0$

Main Body:

We start by checking the activities that immediately succeed activity 1. These activities are 2, 3. Activities 2 and 3 have single predecessor. ES_2 *and* ES_3 are calculated as follows:

$ES_2 = ES_3 = ES_1 + t_1 = 0 + 6 = 6$

Using ES_2 *and* ES_3, we calculate ES_4, ES_5 and ES_7

$ES_4 = ES_5 = ES_2 + t_2 = 6 + 12 = 18$

$ES_7 = ES_3 + t_3 = 6 + 7 = 13$

ES_6 is calculated *using* ES_4

$ES_6 = ES_4 + t_4 = 18 + 5 = 23$

Activity 8 has 3 predecessors. Therefore,

$ES_8 = Max\{ES_6 + t_6; ES_5 + t_5; ES_7 + t_7\} = Max\{23+2; 18+3; 13+9\} = 25$

Since all ES values are calculated, the project duration is calculated

$T = \text{Max}\{ES_i + t_i\} = 25 + 5 = 30; LC_8 = 30$

As a result, the project duration is calculated. Now it is time to calculate the latest completion time values. LC_5, LC_6, LC_7 are calculated using LC_8

$LC_5 = LC_6 = LC_7 = LC_8 - t_8 = 30 - 5 = 25$

Similarly, LC_3 and LC_4, are calculated

$LC_3 = LC_7 - t_7 = 25 - 9 = 16$

$LC_4 = LC_6 - t_6 = 25 - 2 = 23$

Activity 2 has 2 immediate successors. Therefore,

$LC_2 = Min\{LC_5 - t_5; LC_4 - t_4\} = Min\{25 - 3; 23 - 5\} = 18$

Activity 1 has 2 immediate successors.

$LC_1 = \text{Min}\{LC_2 - t_2; LC_3 - t_3\} = \text{Min}\{18 - 12; 16 - 7\} = 6$

All latest completion times and slack times are calculated and tabulated in Table 2.3.

From the table, it can be observed that the slack times and durations of the activities 1, 2, 4, 6 and 8 are equal. These activities are critical, and the path defined by these activities is the critical path. The critical activities in Figure 2.4 are shown by solid lines.

Table 2.3: CPM Calculations of an Example

Activity	Pred.	Succ.	Duration	ES	LC	slack
1	-	2,3	6	0	6	6
2	1	4,5	12	6	18	12
3	1	7	7	6	16	10
4	2	6	5	18	23	5
5	2	8	3	18	25	7
6	4	8	2	23	25	2
7	3	8	9	13	25	12
8	5,6,7	-	5	25	30	5

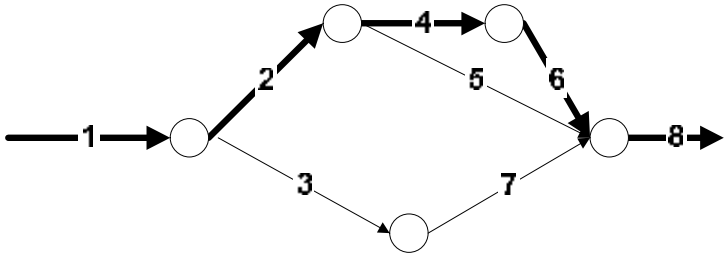

Figure 2.4: Critical Path

Observe that $T = t_1 + t_2 + t_4 + t_6 + t_8 = 30$. Any change in the duration of any critical activity directly affects the project duration. On the other hand, the duration of activity 4 is 7, and the slack time of that activity is 10. This means that the duration of activity could be increased by 3 units without increasing the project duration. In other words, one can delay the start time of activity 4 by 3 units without increasing the project duration.

2.2 Formulations for Multi-Mode Problems

The activity durations can often be reduced by dedicating additional money resources. Money resources are explained as the cost of activities. In the discrete version of the problem, each activity is given a number of time/cost pairs, so called modes. The cost time pair of activity is (c_{ik}, t_{ik}) where c_{ik} is the cost of activity i and t_{ik} is the duration (time) of activity i, associated to its k'th mode. For any two modes $(i,k1)$ and $(i,k2)$, we assume that $t_{ik1} > t_{ik2}$ implies $c_{ik1} < c_{ik2}$, i.e., shorter durations require extra resources, hence higher costs.

Each activity i has m_i modes. The problem is to assign a mode for each activity by considering total cost and project duration as criteria. This problem is referred to as Discrete Time/Cost Trade-off Problem (DTCTP) in the literature. We define three versions of the DTCTP and give their mathematical representations. These problems are Deadline, Budget, and Time/Cost Curve Problems. We give the mathematical representations of the problem. Similar formulations can be found in many project scheduling references.

Mathematical Model:

Indices, parameters and decision variables of the mathematical model, proposed are as follows.

Indices:

i: activity index $\quad i = 1, 2, ..., N+1$

k: mode index $\quad k = 1, 2, ..., m_i$

(N+1 is a dummy activity where all activities having no successors are connected to)

Parameters:

m_i: number of modes for activity i

t_{ik}: duration of activity i with mode k

c_{ik}: cost of activity i with mode k

P_i: set of immediate predecessors of activity i

Decision Variables:

S_i =Starting time of activity i

$$y_{ik} = \begin{cases} 1, \text{ if activity } i \text{ is assigned to mode } k \\ 0, \text{ otherwise} \end{cases}$$

S_{N+1} =Project duration

Constraints:

Each activity should be assigned to exactly one mode.

$$\sum_{k=1}^{m_i} y_{ik} = 1 \qquad i = 1, 2, ..., N \qquad (c1)$$

An activity cannot start before all of its immediate predecessor activities are completed.

The time of the assigned mode for activity i is t_i, where $t_i = \sum_{k=1}^{m_i} t_{ik} \times y_{ik}$

$$S_i \geq S_j + \sum_{k=1}^{m_j} y_{jk} \times t_{jk} \quad i = 1, 2, ..., N+1; \forall j \in P_i \qquad (c2)$$

Starting time of the activities should be nonnegative.

$$S_i \geq 0 \qquad i = 1, 2, ..., N+1 \qquad (c3)$$

y_{ik}'s are binary variables

$$y_{ik} \in \{0,1\} \qquad i = 1, 2, ..., N; k = 1, 2, ..., m_i \qquad (c4)$$

The problem contains $\sum_{i=1}^{N} m_i$ binary y_{ik} variables, and $N+1$ continuous S_i variables. There are $N + \sum_{i=1}^{N} |P_i|$ constraints.

We now define the additional constraints and the objective of each individual problem.

1. Deadline Problem

We find the least costly mode assignments subject to the constraint that the project completes by the specified deadline.

We let t denote this deadline and add the following constraint.

$$S_{N+1} \leq t \qquad (c5)$$

The objective function of the Deadline Problem is the total cost and is expressed as:

$$Min \sum_{i=1}^{N} \sum_{k=1}^{m_i} c_{ik} \times y_{ik}$$

The Deadline Problem model is

$$Min \sum_{i=1}^{N} \sum_{k=1}^{m_i} c_{ik} \times y_{ik}$$

$$Subject\ to\ (c1), (c2), (c3), (c4), (c5)$$

2. Budget Problem

We find the smallest project duration, subject to the constraint that the total cost of the selected modes does not exceed the available budget.

We let *b* denote the available budget and add the following constraint

$$B = \sum_{i=1}^{N} \sum_{k=1}^{m_i} c_{ik} \times y_{ik} \leq b \qquad (c6)$$

The objective function of the Budget Problem is expressed as:

$Min\ S_{N+1} = Min\ T$

i.e., the project duration is to be minimized.

The Budget Problem model is

$Min\ S_{N+1}$

$Subject\ to\ (c1), (c2), (c3), (c4), (c6)$

3. Time/Cost Curve Problem

The problem is to assign a mode for each activity such that a non decreasing function of *T* and *B* is minimized. We let *f (T,B)* denote this function. *f (T,B)* may be known or unknown. An example to *f (T,B)* is a linear combination of *T* and *B*, such that $w_1T + w_2B$ where $w_1(w_2)$ is the relative weight assigned to *T(B)*. When *f (T,B)* is unknown, then the problem reduces to the generation of all efficient solutions with respect to *T* and *B*.

A solution *S* is efficient if there does not exist a schedule S^* such that $T(S^*) \leq T(S)$ and $B(S^*) \leq B(S)$ with strict equality holding at least once. Provided that *f (T,B)* is non-decreasing in *T* and *B*, an optimal solution is in the efficient set. Hence to find an optimal solution to any non-decreasing *f (T,B)*, it is sufficient to generate all efficient solutions, evaluate each solution for *f (T,B)* and select the best solution.

In the literature, the Efficient Set Generation Problem is referred to as Time/Cost Curve Problem. The previous studies solve the Deadline Problem for all possible realizations of the project durations. Alternatively, one can use Budget Problem for all possible realizations of the total cost values. In this thesis, we study the Deadline Problem, and use this problem to generate all efficient solutions, i.e., to solve the Time/Cost Curve Problem.

2.3 Literature Review on Multi-Mode Problems

In this chapter, we explain current studies on Time/Cost Trade-off Problem in chronological order.

The paper by Fulkerson (1961) is one of the first successful attempts to handle the project scheduling problem with time/cost trade-offs. The relation between time and cost of each activity is defined by a single linear function. Activity on Arc (AoA) representation is used and each activity is represented as (*i,j*) by its source node "*i*" and terminal node "*j*". The focused problem is to compute time/cost curve using the Deadline Problem. Fulkerson (1961) converts the Deadline Problem into a Network Flow Problem, by taking its dual and doubling each arc. Fulkerson (1961) proves that the time/cost curve is convex and develops a Network Flow Algorithm.

Kelley (1961) proposes a very similar algorithm to Fulkerson's to derive time/cost curve. He proposes, Primal-Dual Algorithm followed by the Network Flow Algorithm, which is derived from the dual of the main problem. Moreover, Kelley (1961) studies on the structure of the project network and relations of the activities. This study is one of the pioneer studies in network decomposition in project scheduling. Kelley defines the time/cost curve as "Project Cost Utility Curve". As a result, by estimating utility function of the user, Project Cost Curve helps to make conclusions about the solution and conduct sensitivity analysis. Kelley (1961) concludes his work by discussing some real life cases.

Meyer and Shaffer (1965) focus on the Deadline Problem and consider continuous, discontinuous, discrete, combination of discrete and continuous functions. Moreover, they mention convex, concave and hybrid cases of these functions separately and develop Mixed Integer Programming Formulations for each case. They conclude their study with an example of nine-activity project network, where each activity has time/cost relationship of its own. However, no computational work is presented.

Butcher (1967) is the first to handle the Budget Problem with Dynamic Programming Formulation, in which the activities form the stages and the money paid to activities are state variables. The decision is how much to allocate to each activity. The author concludes his work by mentioning the need for further analysis on the networks that cannot be decomposed into serial or parallel structures.

Crowston *et al.* (1967) approach the Discrete Time/Cost Trade-off Problem from the Decision Critical Path Methodology content. The network is represented as a decision tree. The activities are represented via nodes. The paper is one of the first Discrete Time/Cost Trade-off Problem works where Activity on Node representation is used. Their computational results show the superiority of the Mixed Integer Formulation over the Complete Enumeration. The interdependency conditions are added between some decision nodes. The paper models the Deadline Problem by a Mixed Integer Formulation. By removing some redundant constraints, the new formulation named Reduced Path Formulation, is provided. Moreover, a Heuristic Method is proposed which can handle only mutually exclusive type interdependencies.

Rothfarb *et al.* (1970) and Frank *et al.* (1971) work on the optimal design of Offshore Natural-gas Pipeline Systems Problem. They develop algorithms to merge series and parallel connected activities and they determine some rules to set up the modes of the merged activities.

Robinson (1975) develops a Dynamic Programming Algorithm that handles the Budget Problem for all possible network structures with discrete time/cost relations. Robinson (1975) states that the complexity of the problem is reduced if the activity durations are given at uniformly spaced points like: d, 2d, 3d. After defining a general recursive formula, Robinson (1975) defines sufficiency conditions to decompose the recursive formula into One-dimensional Optimization Problem. If the conditions do not hold, the problem is solved as a Multidimensional Allocation Problem. The author proposes mode elimination methods, elimination and addition of the precedence relations to reduce the computational effort of the Multidimensional Problems. To solve the Multidimensional Problem, state variables are chosen and set to constant values to convert the problem into One-dimensional Optimization Problem. Later the problem is solved over different values of the state variables. This paper is the leader in the project network decomposition with the clearly stated sufficiency and necessary conditions. However, the theoretical work is not supported with computational experience and examples.

Phillips *et al.* (1977) work on the Linear Time/Cost Relationship Problem. They improve the algorithms of Fulkerson (1961) and Kelley (1961) for generating the time/cost curve. In addition to Fulkerson (1961) and Kelley (1961)'s algorithms, the proposed algorithm locates the minimal cuts on the network by using flow

interpretations. To locate the minimal cuts, manual inspection, Out of Kilter and Cut Search Algorithms are used. Their computational experience shows that Cut Search Algorithm performs superior to the Out of Kilter Algorithm.

Elmaghraby (1993) addresses the Budget Problem, and proposes a Dynamic Programming approach. He first reduces the network to an s/p reducible network by fixing the activity durations. The reduced problem can be solved as a One-dimensional Optimization Problem as stated by Robinson (1975). Later the sequence of node reductions and arc fixings are determined. Dynamic Programming formulation is used at every step of node reductions and arc fixings. After network reduction is completed, the network is updated and the fixed activities are labeled. The reduced problem is solved by Branch and Bound because of the curse of dimensionality. At each level of the tree a labeled activity is chosen and the decision is which mode (duration) to choose. The Optimization Algorithm is followed by an Approximation Algorithm. All proposed algorithms are illustrated by numerical examples. However, no computational work is presented.

De *et al.* (1995) present an extensive literature review of Time/Cost Trade-off Problems in Project Scheduling. All studies are classified according to the structure of time/cost function and objective function focus. A general definition of all problem types is provided.

Demeulemeester *et al.* (1996) consider all optimal procedures of DTCTP for all problem types: Deadline Problem, Budget Problem, Time/Cost Curve Problem; and overview the solution methodologies. For s/p reducible networks, series parallel merge algorithms developed by Rothfarb *et al.* (1970) is recommended. For s/p irreducible networks a partial enumeration approach is proposed. The main goal of the approach is to convert the network into an s/p reducible network by optimal fixings previously discussed by Elmagraby (1993). A similar network reduction approach to Bein *et al.* (1992) is proposed. One main difference is to use “minimum number of activity fixings”, instead of “minimum number of node reductions” to evaluate reduction complexity. The “minimum number of activity fixings” is defined to be the *Complexity Index*, which is commonly used to evaluate reduction complexity in later studies. The operations during the reduction are coded, and recorded as the *reduction plan*.

For the Time/Cost Curve Problem, the solution is found by complete enumeration of all modes of fixed activities with the help of backtracking of the

reduction plan. For the Budget and Deadline Problems, the solution procedure is based on the same backtracking procedure. Some dominance rules are also included. Their new reduction plan performs better in less complex networks than the reduction plan of Bein *et al.* (1992). However, in complex networks the new algorithm is outperformed.

Dunne *et al.* (1997) prove that all versions of the Discrete Time/Cost Trade-off Problem are NP Hard, in the strong sense, which is the most important complexity result of DTCTP literature. They also show that some special structures like pure parallel, pure series are solvable in polynomial times.

Skutella (1998) addresses approximation algorithms for Discrete Time/Cost Trade-off Problem. Skutella first assigns two modes for each activity, where the shorter is zero. There after, the problem is easily relaxed to a linear problem. Using linear relaxation results feasible discrete realizations are constructed. Skutella (1998) proves that there are approximations with performance guarantee l for the problems with durations range: $\{0, 1, 2, l\}$ for the Budget Problem. For the special case with duration range $\{0, 1, 2\}$, an approximation algorithm with performance guarantee of 3/2 is presented. The results are extended to an Approximation Algorithm with performance guarantee O ($\log l$) where l is the ratio of the maximum duration to the minimum nonzero duration. In the last part, Approximation Algorithms for the Time/Cost Curve Problem with a constant error factor are discussed.

Demeulemeester *et al.* (1998) consider the Time/Cost Curve Problem and solve the problem by a horizon-varying approach using iterative solutions of the Deadline Problem. The Deadline Problems are solved by Branch and Bound Algorithm using Linear Relaxation based lower bounds. They provide piecewise linear approximations for the time/cost function, and solve the Linear Relaxation Problem by Fulkerson (1961)'s algorithm. This adaptation is due to having more than two modes which is not taken into account by Fulkerson. Their algorithm solves the small-sized instances up to 30 activities and 4 modes easily, however fails to solve the majority of the instances with 40 activities.

Deineko *et al.* (2001) prove that there cannot exist a polynomial time approximation algorithm with a performance guarantee better than 3/2 for any versions of the Discrete Time/Cost Trade-off Problem. They also prove that there cannot exist a polynomial time $(1+\varepsilon, 5/4-\beta)$-Approximation Algorithm for any

versions of the Discrete Time/Cost Trade-off Problem; given that there exist a real number $\varepsilon>0$ and $\beta>0$.

Akkan *et al.* (2005a) develop heuristics and lower bounding procedures for the Deadline Problem. Similar to the previous approaches, they compute linear relaxation based lower bounds. Their first bound uses cuts based on the earliest start and latest completion times of the activities in their linear program. Their second lower bounding procedure uses network decomposition as the basic idea. After a number of sub networks are found, the mathematical model is re-formulated with the inclusion of the sub networks. The LP Relaxation of the reformulation gives a lower bound and it is solved by column generation procedure. The proposed heuristic also depends on the column generation based realizations of the Deadline Problem. They propose two rules to eliminate short and long modes. Their extensive computational study reveals the satisfactory behavior of their algorithm.

The most closely related study to ours is due to Akkan *et al.* (2005a). We propose optimization algorithm and heuristic procedures for the Deadline Problem whereas Akkan *et al.* (2005a) propose bounding procedures. Moreover, we extend our findings for Deadline Problem to Time/Cost Curve Problem.

CHAPTER 3

THE DEADLINE PROBLEM

The Deadline Problem finds the least costly project schedule with a given project deadline. We develop a Branch and Bound Algorithm to solve the Deadline Problem. In this section we first discuss procedures used to reduce the problem size. Next lower bounding procedures are discussed that are used in the Branch and Bound Algorithm. Finally, we present our upper bounding procedures.

3.1 Problem Size Reduction

We develop some rules that help us to reduce the problem size. These reductions are done through activity-mode eliminations. Such eliminations are valuable in the sense that they reduce the number of binary variables, thereby dispelling the exponential nature of the problem, to some extent.

We develop two elimination rules: one for discarding short modes and one for long modes.

We use the following notation to state our rules.

LC_j(S): latest completion time of activity j when the shortest activity durations are used.

ES_j(S): earliest start time of activity j when the shortest activity durations are used.

LC_j(L): latest completion time of activity j when the longest activity durations are used.

ES_j(L): earliest start time of activity j when the longest activity durations are used.

t_j: the duration of the mode assigned to activity j.

$$t_j = \sum_{k=1}^{m_j} t_{jk} \times y_{jk}$$

c_j: the cost of the mode assigned to activity j.

$$c_j = \sum_{k=1}^{m_j} c_{jk} \times y_{jk}$$

We now give the formal statements of our rules. Theorem 1 eliminates long modes whereas Theorem 2 eliminates short modes.

Theorem 1: If $t_{jk} \geq LC_j(S) - ES_j(S) + 1$ then $\sum_{r=1}^{k} y_{jr} = 0$, in all feasible solutions.

Proof: Note that $LC_j(S) - ES_j(S)$ is the maximum allowable processing time for activity *j*, as the other activities are set to their smallest activity durations. Hence any activity duration greater than $LC_j(S) - ES_j(S)$ returns an infeasible solution. Therefore y_{jr} should be set to zero, if $t_{jk} \geq LC_j(S) - ES_j(S) + 1$. If $t_{jk} \geq LC_j(S) - ES_j(S) + 1$ then $t_{jr} \geq LC_j(S) - ES_j(S) + 1$ if $r \leq k$, as, $t_{jr} \geq t_{jk}$. This follows, $y_{j,k} = y_{j,k-1} = ... = y_{j,1} = 0$, i.e., $\sum_{r=1}^{k} y_{jr} = 0$. □

Akkan *et al.* (2005a) give the same result for the last mode m_j. Hence Theorem 1 is a simple generalization of Akkan *et al.* (2005a)'s theorem.

Theorem 2: If $t_{jk} < LC_j(L) - ES_j(L)$ then $\sum_{r=k+1}^{m_j} y_{jr} = 0$, in all optimal solutions.

Proof: Assume a solution that contradicts with the condition of the theorem, i.e., $t_{jk} < LC_j(L) - ES_j(L)$ and y_{jr}=1 for any $r \geq k+1$.

Assume the mode of activity *j* is changed from *r* to *k*. Such an exchange is feasible as $t_{jk} < LC_j(L) - ES_j(L)$. (Note that any processing smaller than or equal to $LC_j(L) - ES_j(L)$ will not increase project duration, hence never causes infeasibility), and it decreases the objective function by $c_{jr} - c_{jk} > 0$ units. This implies a solution that contradicts with the condition of the theorem cannot be optimal. □

We implement Theorem 1 and Theorem 2 successively, once Theorem 1 eliminates some modes, the longest feasible modes do change and the conditions stated by Theorem 2 will hold with higher probability. Similarly once Theorem 2 eliminates some short modes, the $LC_j - ES_j$ values will decrease, and the probability that Theorem 1 eliminates becomes higher.

We check the theorems alternately and stop when neither of them can eliminate any mode. Below is the stepwise description of our mode elimination procedure.

Step 0: a=1

Step 1: Find, $LC_j(S)$, $ES_j(S)$ using critical path method by setting shortest activity times to each activity.

Step 2: Find smallest k that satisfies $t_{jk} \geq LC_j(S) - ES_j(S) + 1$.

If no such k exists and a=0, then stop.

Else Eliminate modes 1,…,k. Set $m_j = m_j - k$.

Step 3: a=0

Find, $LC_j(L)$, $ES_j(L)$ using critical path method by setting longest activity times to each activity.

Step 4: Find largest k that satisfies $t_{jk} < LC_j(L) - ES_j(L)$.

If no such k exists and a=0, then stop.

Else Eliminate modes $k+1$,…,m_j. Go to Step 1.

Numerical Example

Consider the following 5-activity network shown in Figure 3.1, where activities 1, 2, 4 and 5 have three modes and activity 3 has two modes. The time/cost pairs are given on the arcs of the figure.

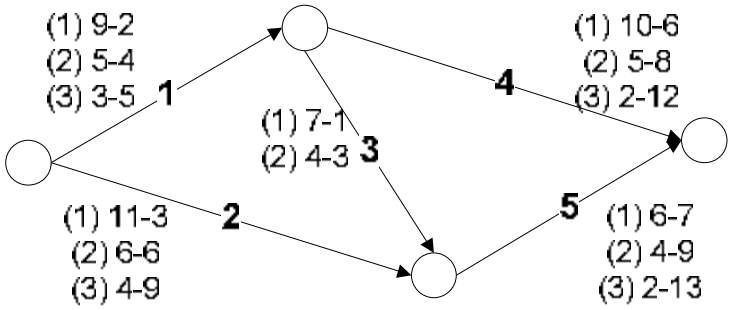

Figure 3.1: AoA Representation of the Sample Project Network

Based on the given data, the minimum and maximum possible durations are 9 and 22 respectively. The problem is first solved for deadline value of 11.

Step 0: a=1

Step 1: All activities are set to their shortest activity durations. Hence,

$t_1 = 3; t_2 = 4;\ t_3 = 4;\ t_4 = 2;\ t_5 = 2.$

Table 3.1 below shows CPM calculations with t=11

Table 3.1: CPM Calculations Summary with Shortest Activity Durations (Iteration 1 Results with t =11)

Activity	Pred.	Succ.	Duration	ES(S)	LC(S)
1	-	3,4	3	0	5
2	-	5	4	0	9
3	1	5	4	3	9
4	1	-	2	3	11
5	2,3	-	2	7	11

Step 2: Following mode eliminations can be done:

$LC_1(S) - ES_1(S) + 1 = 6 < t_{1,1} = 9;$ Mode (1,1) is eliminated.

$LC_2(S) - ES_2(S) + 1 = 10 < t_{2,1} = 11;$ Mode (2,1) is eliminated.

$LC_3(S) - ES_3(S) + 1 = 7 = t_{3,1} = 7;$ Mode (3,1) is eliminated.

$LC_4(S) - ES_4(S) + 1 = 9 < t_{4,1} = 10;$ Mode (4,1) is eliminated.

$LC_5(S) - ES_5(S) + 1 = 5 < t_{5,1} = 6;$ Mode (5,1) is eliminated.

A total of 5 modes are eliminated. This reduces the number of the binary variables from 162 ($3\times3\times2\times3\times3$) to 16($2\times2\times1\times2\times2$).

Step 3: a=0

Now, all activities are set to their longest activity durations. Table 3.2 below shows CPM calculations.

Table 3.2: CPM Calculations Summary with Longest Activity Durations (Iteration 1 Results with *t* =11)

Activity	Pred.	Succ.	Duration	ES(L)	LC(L)
1	-	3,4	5	0	3
2	-	5	6	0	7
3	1	5	4	5	7
4	1	-	5	5	11
5	2,3	-	4	9	11

Step 4: Note that activity 3 has a single mode left. Hence, it is not considered in this step.

$LC_1(L)-ES_1(L)=3;$ Activity 1 cannot be shorter.

$LC_2(L)-ES_2(L)=7>t_{2,2}=6;$ Mode (2,3) is eliminated.

$LC_4(L)-ES_4(L)=6>t_{4,2}=5;$ Mode (4,3) is eliminated.

$LC_5(L)-ES_5(L)=2;$ Activity 5 cannot be shorter.

We eliminate two more modes and return to Step 1.

Step 1: At this step the remaining modes are second modes of all activities and third modes of activity 1 and 5. Activities with more than one mode are 1 and 5. Both activities are set to their shortest activity duration. Hence, $t_1=3; t_2=6;\ t_3=4;$ $t_4=5;\ t_5=2$. Table 3.3 below shows the CPM calculations.

Table 3.3: CPM Calculations Summary with Shortest Activity Durations (Iteration 2 Results with *t* =11)

Activity	Pred.	Succ.	Duration	ES(S)	LC(S)
1	-	3,4	3	0	5
2	-	5	6	0	9
3	1	5	4	3	9
4	1	-	5	3	11
5	2,3	-	2	7	11

$LC_1(S) - ES_1(S) + 1 = 6$; Activity 1 cannot be longer.

$LC_5(S) - ES_5(S) + 1 = 5$; Activity 5 cannot be longer.

No further elimination can be done; a=0. We stop here. Therefore, the number of binary variables is reduced to 4 ($2 \times 1 \times 1 \times 1 \times 2$) if the deadline is set to 11.

Now consider the deadline of 18.

Step 0: a=1

Step 1: All activities are set to their shortest activity durations. Hence, $t_1 = 3; t_2 = 4;\ t_3 = 4;\ t_4 = 2;\ t_5 = 2$. Table 3.4 below shows CPM calculations with t=18.

Step 2: Mode elimination calculations are done as follows:

$LC_1(S) - ES_1(S) + 1 = 13;\ LC_2(S) - ES_2(S) + 1 = 17;$

$LC_3(S) - ES_3(S) + 1 = 14;\ LC_4(S) - ES_4(S) + 1 = 16;$

$LC_5(S) - ES_5(S) + 1 = 12$

There is not any possible mode elimination at this step. We proceed to Step 3.

Table 3.4: CPM Calculations Summary with Shortest Activity Durations (Iteration 1 Results with *t* =18)

Activity	Pred.	Succ.	Duration	ES(S)	LC(S)
1	-	3,4	3	0	12
2	-	5	4	0	16
3	1	5	4	3	16
4	1	-	2	3	18
5	2,3	-	2	7	18

Step 3: a=0

All activities are set to their longest activity durations. Table 3.5 below shows CPM calculations.

Table 3.5: CPM Calculations Summary with Longest Activity Durations (Iteration 1 Results with *t* =18)

Activity	Pred.	Succ.	Duration	ES(L)	LC(L)
1	-	3,4	9	0	5
2	-	5	11	0	12
3	1	5	7	9	12
4	1	-	10	9	18
5	2,3	-	6	16	18

Step 4: Mode elimination calculations are done as follows:

$LC_1(L) - ES_1(L) = 5 = t_{1,2}$; Mode (1,3) is eliminated.

$LC_2(L) - ES_2(L) = 11 > t_{2,2} = 6$; Mode (2,3) is eliminated.

$LC_3(L) - ES_3(L) = 2$; Activity 3 cannot be shorter.

$LC_4(L) - ES_4(L) = 8 > t_{4,2} = 5$; Mode (4,3) is eliminated.

$LC_5(L) - ES_5(L) = 1$; Activity 5 cannot be shorter.

Three modes are eliminated. We return to Step 1.

Step 1: All modes are set to their shortest activity durations. Table 3.6 below shows the CPM calculations.

Earliest start and latest completion values do not change. No elimination can be done, a=0, and hence we stop.

Therefore, the number of binary variables is reduced to 48 ($2\times2\times2\times2\times3$).

Table 3.6: CPM Calculations Summary with Shortest Activity Durations (Iteration 2 Results with t =18)

Activity	Pred.	Succ.	Duration	ES(S)	LC(S)
1	-	3,4	5	0	12
2	-	5	6	0	16
3	1	5	4	5	16
4	1	-	5	5	18
5	2,3	-	2	7	18

When deadline is set to 11, the mode elimination algorithm eliminates long modes first, then short modes and terminates. When the deadline is set to 18 the mode elimination algorithm cannot eliminate any long modes but one more short mode. The results are intuitive as the deadlines closer to minimum possible durations helps to eliminate more long modes, whereas the deadlines that are closer to maximum possible duration helps to eliminate more short modes.

3.2 Lower Bounds

We develop two lower bounding procedures on the optimal total budget value. These are namely Naive Bound and Linear Programming (LP) Relaxation Based Lower Bound.

3.2.1 Naive Bound, LB_1

LB_1 is found by assigning each activity to its minimum cost mode. Hence, LB_1 is simply $\sum_{i=1}^{N} c_{i,1}$, where the modes are in their ascending order of costs. The resulting solution is optimal, if the deadline constraint is satisfied.

In our experiments, we use LB_1 as a filtering mechanism. We first compare LB_1 evaluated at a particular node, with the best known solution value, Z_{inc}. If $LB_1 \geq Z_{inc}$ we fathom the node, otherwise we calculate the LP Relaxation based lower bound.

In calculating naive lower bound, we first make mode eliminations, due to the long modes, then select the smallest cost among the remaining modes.

3.2.2 LP Relaxation Based Bound, LB_2

LB_2 is found simply by relaxing the integrality constraints on y_{ik} values and letting $0 \leq y_{ik} \leq 1$ for all j and k. We calculate LP Relaxation based LB after reducing size of the sub problem at a node, by mode elimination rules. We show that mode eliminations will increase the value of the lower bound without causing infeasibility.

Assume that we have a solution space as shown in Figure 3.2**.**

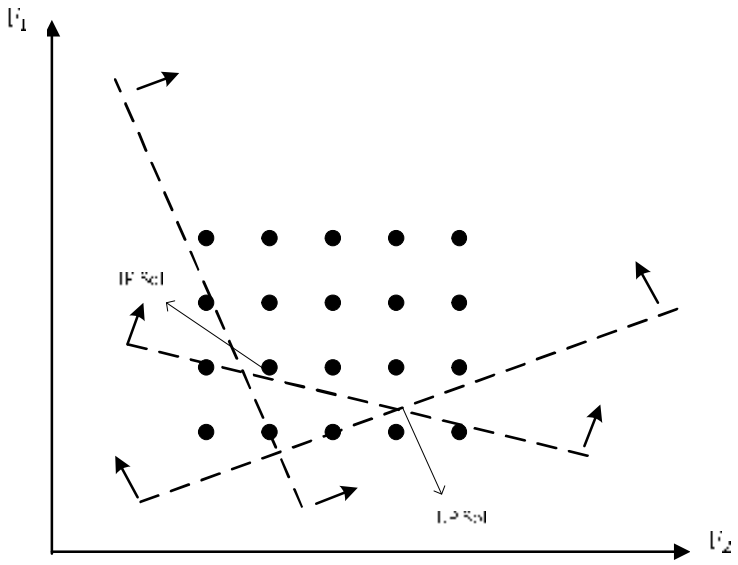

Figure 3.2: Sample Solution Space

We have 3 constraints and have an arbitrary minimization function over F_1 and F_2. The optimal solution and the LP relaxation solution are as shown in the figure. Mode elimination algorithms reduce solution space in a way that the discarded modes are never used. Note that, both mode elimination procedures eliminate successive modes for each activity. In the figure below, mode elimination addition is represented by an additional line.

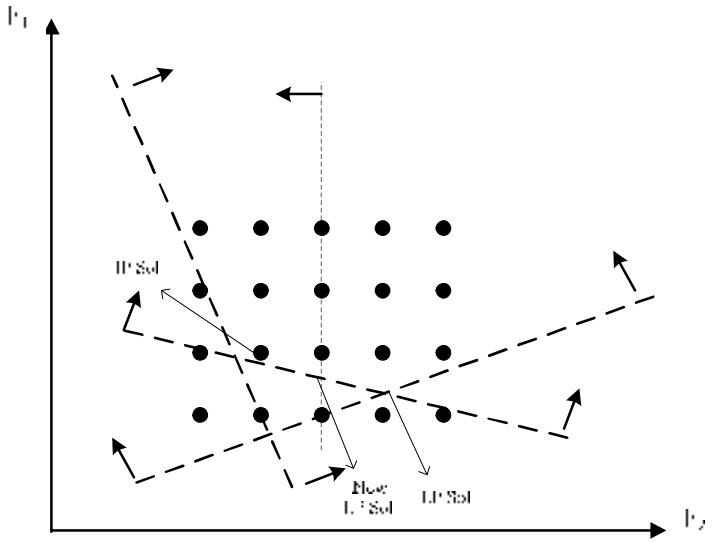

Figure 3.3: Sample Solution Space with Mode Elimination

Note that, after mode eliminations, the new solution has no smaller objective function value as additions of new constraints never improve. Hence the resulting lower bound is stronger.

Properties of LP Relaxation Solutions

In this section, we present two properties of the optimal LP solutions. We hereafter refer to t_i^{LP} as the optimal duration of activity *i* in the LP Relaxation solution. We let c_i^{LP} denote the associated cost of activity *i*. The following theorem proves that at most two modes of an activity can take positive values in the optimal LP Relaxation solution.

Theorem 3: There exists an optimal LP Relaxation Problem in which $y_{ik}>0$ for at most two modes, for each activity.

Proof: The minimum cost t_i^{LP}, i.e. the duration found by the optimal LP relaxation, is available through the following LP Problem.

(P_0) $\quad Min \sum_{k=1}^{m_i} c_{ik} \times y_{ik}$

Subject to

$$\sum_{k=1}^{m_i} t_{ik} \times y_{ik} = t_i^{LP}$$

$$\sum_{k=1}^{m_i} y_{ik} = 1$$

$$y_{ik} \geq 0$$

Note that (P_0) has only two constraints. Therefore, every basic feasible solution has two basic variables. That is, there are at most two positive y_{ik} values, in all basic feasible solutions. From the LP theory, we know that there is an optimal solution which is basic feasible solution.

Hence an optimal solution to the LP Relaxation gives at most two positive, i.e., fractional values, for each activity *i*. □

The following theorem states that, in the optimal LP Relaxation solution if an activity is non-critical then it is assigned to the highest duration mode.

Theorem 4: If activity *i* is non-critical in optimal LP Relaxation solution then $t_i^{LP} = t_{i,1}$ and $c_i^{LP} = c_{i,1}$.

Proof: In an optimal LP relaxation solution, t_i^{LP} cannot be increased due to the following two reasons:

1. $t_i^{LP} = t_{i,1}$, i.e., no further increase is possible
2. Any further increase violates feasibility.

If activity *i* is non-critical then any further increase does not violate feasibility. As t_i^{LP} cannot be increased for optimal solution and any further increase does not violate feasibility then t_i should be $t_{i,1}$, i.e., as case 2 does not hold, case 1 should hold. □

Theorem 4 implies that if activity *i* is non-critical then $t_i^{LP} = t_{i,1}$. However, the reverse does not necessarily hold, i.e., a critical activity *i* may also have $t_i^{LP} = t_{i,1}$

3.3 Branch and Bound Algorithm

We use the result of Theorem 3 to define our branching structure. At every branch, LP Relaxation Problem is solved. We always branch from a fractional variable of the LP Relaxation solution. For the chosen fractional value y_{ik}, i.e., $0 < y_{ik} < 1$, we generate the following sub problems.

Subproblem 1: $y_{ik} = 0$

Subproblem 2: $y_{ik} = 1$

The associated tree is shown in Figure 3.4 below.

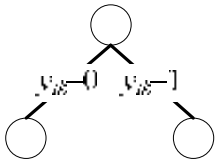

Figure 3.4: Branching Scheme

We know discuss our selection strategies.

Selection Strategies

We employ the following three strategies to select the fractional variable from which two sub problems are generated.

Strategy 1: Select the highest y_{ik} value.

Strategy 2: Select the highest y_{ik_i} value where k_i satisfies $k_i = Min\{k \mid y_{ik} > 0\}$, i.e., for each partially assigned activity, the smallest cost partial assignment is selected.

Strategy 1 expects that the optimal integer solution is close to the optimal LP Relaxation cost, hence forces the big y_{ik} values to 1, in earlier branches. ***Strategy 2*** considers the objective function of minimizing total cost, and selects the highest fractional value among the smallest cost choices. ***Strategy 3***, below, considers the cost differences between the consecutive modes. Among the smaller cost choices of the fractional activities, it identifies the mode with the highest cost difference with its consecutive mode. Hence, mode selections that result in high total cost increase are avoided.

Strategy 3: Select the y_{ik_i} value with the highest "$c_{i,k_i} - c_{i,k_i+1}$" value where k_i satisfies $k_i = Min\{k \mid y_{ik} > 0\}$, i.e., for each activity the smallest cost alternative is evaluated. Among those alternatives, the mode with the highest cost difference with its successive mode is selected.

We illustrate the selection strategies via the following numerical example instance. Consider the 15-activity problem instance whose data are given in Table 3.7 below.

Table 3.7: Modes and Precedence Relations for Example Problem

Activity	Successors	Modes
1	3, 4, 5	16-2; 13-13; 12-16; 11-17; 10-31; 7-35; 4-47; 2-70
2	6, 7	13-6; 10-38; 7-45; 5-52; 3-96
3	9, 10	7-38; 4-71; 3-85
4	13	13-6; 10-38; 7-45; 5-52; 3-96
5	8	17-10; 15-24; 13-28; 10-39; 8-40; 7-77; 5-94; 3-98
6	12	11-7; 10-19; 7-35; 6-57; 4-67; 1-71
7	11, 14	10-15; 7-16; 5-25; 2-75
8	13	12-11; 10-27; 7-35; 6-65; 4-81; 1-90
9	13	15-13; 14-29; 11-41; 8-54; 5-75; 4-77; 2-97
10	15	7-38; 4-71; 3-85
11	12	4-52; 1-73
12	15	17-22; 15-27; 14-30; 11-35; 8-36; 5-37; 2-99
13	15	2-32
14	15	19-39; 17-55; 15-57; 12-63; 10-69; 7-72; 6-86; 3-97
15	16	21-8; 18-19; 15-22; 14-30; 11-33; 8-42; 5-62; 4-73; 2-76

Modes are separated with semicolons. First number is the duration of the mode and second number is the cost of the mode. For instance, activity 4 has a single successor and five modes. The second mode of this activity has duration of 10 and cost of 38.

We set the deadline to 20. Table 3.8 below, shows the fractional variables, and objective function values of LB_2.

Table 3.8: The Optimal LP Solution, i.e., LB_2

Objective Function Value: 673.316667	
Fractional Variables	
$y_{6,1}$	0.700
$y_{6,6}$	0.300
$y_{8,3}$	0.833
$y_{8,6}$	0.166
$y_{9,4}$	0.750
$y_{9,6}$	0.250
$y_{11,1}$	0.667
$y_{11,2}$	0.333
$y_{14,4}$	0.200
$y_{14,6}$	0.800

When ***Strategy 1*** is applied, $y_{8,3}$ is selected as 0.833 is the maximum fractional value.

When ***Strategy 2*** is applied, we consider $y_{6,1}$, $y_{8,3}$, $y_{9,4}$, $y_{11,1}$ and $y_{14,4}$, since only small cost alternatives are considered for all activities. We select the one having the highest value, i.e., $y_{8,3}$=0.833.

When ***Strategy 3*** is applied, we again consider small cost modes. The selection is made from $y_{6,1}$, $y_{8,3}$, $y_{9,4}$, $y_{11,1}$ and $y_{14,4}$. We should calculate cost differences from consecutive modes. Following calculations are done:

$$c_{6,2} - c_{6,1} = 19 - 7 = 12$$
$$c_{8,4} - c_{8,3} = 65 - 35 = 30$$
$$c_{9,5} - c_{9,4} = 75 - 54 = 21$$
$$c_{11,2} - c_{11,1} = 73 - 52 = 21$$
$$c_{14,5} - c_{14,4} = 33 - 30 = 3$$

$y_{8,3}$ is selected since it has the highest cost difference.

Alternate Branching Scheme

We know that in the optimal LP Relaxation solution we have at most two fractional variables for each activity (See Theorem 3). An Alternate Branching Scheme is proposed using this property of the LP Relaxation. At each level an activity with fractional modes is selected. First two nodes assign the activity to the fractional modes. The third node assigns the activity to one of the non-fractional modes, i.e., zero assignment modes. In other words, third node does not allow the

activity to be assigned to one of fractional modes. Assume that $y_{i,k1}$ and $y_{i,k2}$ are fractional variables. The following figure demonstrates the Alternate Branching Scheme.

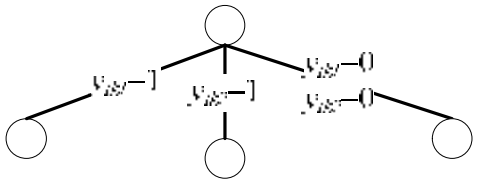

Figure 3.5: Alternate Branching Scheme

One exceptional case for Alternate Branching Scheme is the activities with two modes. In this case third node does not exist.

Alternate Branching Scheme selects activities instead of modes. After the mode is selected by the above selection strategies, the activity of the associated mode is branched, i.e., same selection strategies are used.

3.4 Upper Bounds

In this section, we discuss our heuristic procedures used for obtaining upper bounds. The upper bounds are either used as an initial feasible solution in our Branch and Bound Algorithm or as an heuristic solution for the problem. We classify our upper bounds in two groups:

1. LP Relaxation Based Heuristics
2. Branch and Bound Based Heuristics

LP Relaxation Based Heuristics run in polynomial time, and hence are used as initial feasible solutions. On the other hand, Branch and Bound Based Heuristics run in exponential time and provide heuristic solutions. We now explain the heuristics in detail.

LP Relaxation Based Heuristics

Our LP Relaxation Based Heuristic has two phases. Phase 1 constructs an initial solution. The solution is obtained by modifying the optimal LP Relaxation solution. Improvement Heuristic uses two steps. The first step shifts the modes of

the activities, whereas the second step makes interchanges between the modes of different activities.

Construction Heuristic

The construction phase first finds LP Relaxation solution, then maintains the fractional values without violating the deadline constraint. Note that the duration of an activity in the LP relaxed solution is t_i^{LP} where

$$t_i^{LP} = \sum_{k=1}^{m_i} t_{ik} \times y_{ik}$$

Recall that, the LP relaxed solution has at most two fractional modes for each activity. Hence, t_i^{LP} is the weighted average of the fractional modes, say $k1$ and $k2$. Note that $0 < y_{i,k1} < 1$ and $0 < y_{i,k2} < 1$, and t_i^{LP} is found as follows:

$$t_i^{LP} = t_{i,k1} \times y_{i,k1} + t_{i,k2} \times y_{i,k2}$$

Decreasing t_i^{LP} will increase the total cost however does not increase the project duration. Hence, if we set $y_{i,k}$ to one, where $t_{i,k} < t_i^{LP}$, the feasibility will be preserved. To obtain smaller cost solution we let $y_{i,k} = 1$ if k is the largest processing time mode for activity i, that is no larger than t_i^{LP}. We refer the solution value of this heuristic as UB_1. The stepwise description of UB_1 is given below.

Select all fractional activities. Calculate LP relaxation durations

$t_i^{LP} = \sum_{k=1}^{m_i} t_{ik} \times y_{ik}$ $\qquad i = 1, 2, ..., N$

For $i \in$ *Fractional Activities*

k=1

If $(t_{ik} \le t_i^{LP})$ ***Then***

Select the mode; $y_{ik} = 1$

Else

k=k+1

End If

End For

Calculate upper bound value $UB_1 = \sum_{k=1}^{m_i} c_{ik} \times y_{ik}$

Improvement Heuristic 1

Improvement Heuristic 1 takes the feasible solution by UB_1 and tries to improve. The solution can be improved if the mode of any activity can be increased without violating feasibility. We move activity j from its current mode k_j to mode k_{j-1} if such a movement yields the maximum reduction over all activities. That is, we set $y_{i,k_j-1}=1$ if $c_{j,k_j}-c_{j,k_j-1}=\underset{i}{Max}\left\{c_{i,k_i}-c_{i,k_i-1}\right\}$. We repeat the procedure until any further increase in activity durations results in an infeasible solution. We let UB_2 denote the total cost of the Improvement Heuristic 1. Below is the stepwise description of the heuristic.

$PI=\{1,2,....,N\}$

Remove i from PI , ***If (*** $y_{i,1}=1$***)*** $\quad i=\{1,2,....,N\}$

Calculate improvements $IA_i=c_{i,k}-c_{i,k-1}$ for k's such that $y_{i,k}=1$

Repeat

Select the activity with maximum improvement; $J=Argmax\{IA_i\}$

Change modes. Check feasibility; ***Call (***Critical Path Method***)***

If ($T\le t$ ***) Then***

Let; $y_{J,k}=0$; $y_{J,k-1}=1$

If ($k-1=1$***) Then***

Remove J from PI.; $PI=PI/\{J\}$

Else

Update improvement amount; $IA_i=c_{i,k-1}-c_{i,k-2}$

End If

Else

Let; $y_{J,k}=1$; $y_{J,k-1}=0$

Remove J from PI.; $PI=PI/\{J\}$

End If

Until $PI=\varnothing$

Calculate upper bound value $UB_2=\sum_{k=1}^{m_i}c_{ik}\times y_{ik}$

Note that we ignore the activities that are already set to their minimum cost alternatives.

Improvement Heuristic 2

Recall that, it is not possible to increase the activity duration of any activity in Improvement Heuristic 1 solution, without violating feasibility. However, an improved feasible solution can be obtained by changing modes of two activities simultaneously. Such a change may lead to a feasible solution, provided that the activity duration is decreased while the other activity's processing time is increased and the activities are on the same path. We let *Imp(i,j)* be the amount of improvement obtained if k_i is increased by one (duration is decreased) and k_j is decreased by one (duration is increased). Note that *Imp(i,j)* is not necessarily equal to *Imp(j,i)*. We let $k_r = k_r + 1$ and $k_s = k_s - 1$ if the improvement by activities *r* and *s* is maximum over all pairs *(i,j)*. That is, $\text{Imp(r,s)} = \underset{(i,j)}{Max}\left(c_{i,k_i} - c_{i,k_i+1}\right) + \left(c_{j,k_i} - c_{j,k_i+1}\right)$. We terminate when *Imp(r,s)* is nonpositive. We let UB_3 denote the total cost of the resulting solution. Below is the stepwise description of the Improvement Heuristic 2.

A numerical example that illustrates the implementation of Construction and Improvement Heuristics is given in APPENDIX A.

Calculate improvements as follows:

Imp(i,j)=$\left(c_{i,k}-c_{i,k+1}\right)+\left(c_{j,m}-c_{j,m-1}\right)$ for k's such that $y_{i,k}=1; k \neq m_i$;

for m's such that $y_{j,m}=1; m \neq 1$;

i and j are on the same path

Imp(i,j) =0 $k = m_i$; $m = 1$; i and j are not on the same path

Let Imp(r,s)= $\underset{(i,j)}{Max}\{\text{Imp(i,j)}\}$

Repeat

Change modes;

Check feasibility; ***Call (***Critical Path Method***)***

If ($T \leq t$) ***Then***

$y_{r,k_r}=0; y_{r,k_r+1}=1;\ y_{s,k_s}=0; y_{s,k_s-1}=1$

Recalculate Imp(i,j) for all i,j

Else

Imp(r,s) =0

End If

Let Imp(r,s)= $\underset{(i,j)}{Max}\{\text{Imp(i,j)}\}$

Until Imp(r,s) >0

Calculate upper bound value $UB_3 = \sum_{k=1}^{m_i} c_{ik} \times y_{ik}$

Branch and Bound Based Heuristics

We develop two Branch and Bound Based Heuristics. The first one uses the idea given in Theorem 4. The second heuristic relies on the Alternate Branching Scheme.

Heuristic Branch and Bound Algorithm 1

Theorem 4 states that a non-critical activity in the LP Relaxation solution, should be set to its first mode, i.e., $t_i^{LP} = t_{i,1}$. Heuristic Branch and Bound Algorithm 1 finds non-critical activities at each node and sets them to their first mode, i.e., $t_{i,1}$. As, the non-critical activities in the LP Relaxation solution may be critical in the optimal integer solution, the algorithm cannot ensure optimality. A non-critical

activity in the LP Relaxation is guaranteed to be non-critical in the integer solution only when all successors and predecessors of the activity are non-critical activities, as well.

Heuristic Branch and Bound Algorithm 2

Recall that, Alternate branching scheme has the following structure;

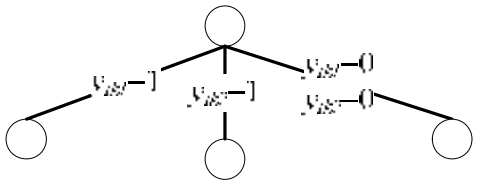

Figure 3.6: Alternate Branching Scheme

The fractional variables $y_{i,k1}$ and $y_{i,k2}$ are taken from the optimal LP Relaxation solution. Note that the first two branches assign the activity modes to one of the fractional values. The third branch searches the other solutions, i.e., $\sum_{k \neq k1,k2}^{m_i} y_{ik} = 1$. If we change the condition of the third branch to $\sum_{k=k1+1}^{k2-1} y_{ik} = 1$ then the resulting solution will ignore the modes that are outside the range [k_1, k_2]. This implies such a change may lead to a nonoptimal solution. However, the resulting solution will be obtained quicker. Figure 3.7 below shows the branching scheme of the Heuristic Branch and Bound Algorithm 2 for a certain level, where the mode of activity i is decided. In the optimal LP solution $0 < y_{i,k1} < 1$ and $0 < y_{i,k2} < 1$.

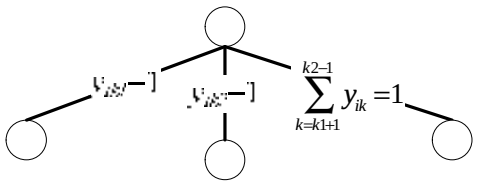

Figure 3.7: Heuristic Branch and Bound Algorithm Scheme

CHAPTER 4

THE TIME/COST CURVE PROBLEM

In this section, we discuss our work on the efficient solution generation for the total cost (B) and project duration (T) criteria. We refer to Efficient Solution Generation Problem as Time/Cost Curve Problem.

We let X denote the set of feasible solutions to our problem. Figure 4.1 illustrates the images of all solutions in the objective space (T,B).

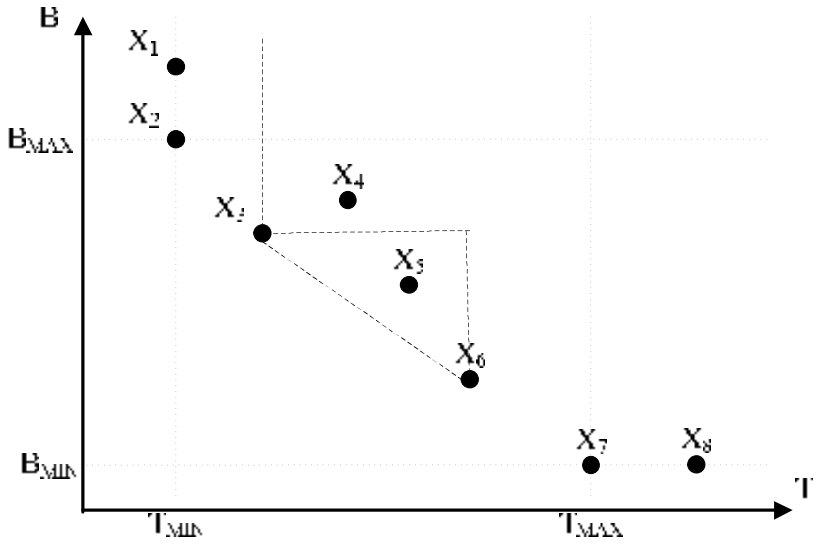

Figure 4.1: Particular Solutions to Time/Cost Curve Problem

In the figure X_2, X_3, X_5, X_6 and X_7 are efficient solutions as they are not dominated by any other solution. X_4 is inefficient as it is dominated by X_3. X_1 and X_2 are minimum project duration solutions. X_1 is dominated by efficient point X_2 as $T(X_2)=T(X_1)$ and $B(X_2)\leq B(X_1)$. X_7 and X_8 are the minimum cost solutions. X_8 is dominated by efficient point X_7 as $B(X_7)=B(X_8)$ and $T(X_7)\leq T(X_8)$. We let X_2 and X_7

as boundary efficient solutions as they optimize a single criterion, T and B respectively.

Note that, B_{MAX} and T_{MAX} are the maximum total cost and project duration values for efficient solutions. Hence, they are upper bounds on the associated criterion values of the efficient solutions. Similarly B_{MIN} and T_{MIN} are the lower bounds on the total cost and project duration values of all efficient solutions.

To generate boundary efficient points, one may use the following single criterion problems.

(P_1) *Min B*

Subject to

$$\sum_{k=1}^{m_i} y_{ik} = 1 \qquad i = 1, 2, ..., N \qquad (c1)$$

$$S_i \geq S_j + \sum_{k=1}^{m_j} y_{jk} \times t_{jk} \qquad \forall j \in P_i ; i = 1, 2, ..., N \qquad (c2)$$

$$S_i \geq 0 \qquad i = 1, 2, ..., N \qquad (c3)$$

$$y_{ik} \in \{0,1\} \qquad i = 1, 2, ..., N ; k = 1, 2, ..., m_i \qquad (c4)$$

(P_2) *Min T*

Subject to $c1, c2, c3$ *and* $c4$

X_1 and X_2 are optimal solutions for (P_2) whereas X_7 and X_8 optimize (P_1). To generate the efficient point with minimum total cost, i.e., B_{MIN}, the following problem has to be solved.

$(P_1)'$ *Min T*

Subject to $B = B_{MIN}$
$c1, c2, c3$ *and* $c4$

where B_{MIN} is the optimal B value of (P_1). An optimal solution to $(P_1)'$ is efficient; this solution is X_2, as it is the minimum completion time solution among the optimal total cost solutions.

Note that for a sufficiently small $e > 0$, $(P_1)'$ is equivalent to $(P_1)''$

$(P_1)''$ $Min\ B + e_T T$

$Subject\ to$

$c1, c2, c3\ and\ c4$

Theorem 5 defines a range for e_T, that equates $(P_1)'$ and $(P_1)''$

Theorem 5: $(P_1)'$ and $(P_1)''$ are equivalent provided that $e_T < \frac{1}{T_{MAX} - T_{MIN}}$.

Proof: e_T is the coefficient of T in the objective function of $(P_1)''$. e_T should be small enough so that B does not change for any change in the value of T. Since it takes only integer values, the minimum change in the B value is 1. Maximum change in T value is $T_{MAX} - T_{MIN}$. Hence, $e_T \cdot (T_{MAX} - T_{MIN}) < 1$, i.e., $e_T < \frac{1}{T_{MAX} - T_{MIN}}$ should hold. □

We set $e_T = \frac{1}{T_{MAX} - T_{MIN} + 1}$, hence solve the $Min\ B + \frac{1}{T_{MAX} - T_{MIN} + 1} T$ problem to generate a boundary efficient point, X_7.

The solution to $(P_1)''$ is (B_{MIN}, T_{MAX}), where T_{MAX} is the maximum total cost value of all efficient solutions, i.e., an upper bound on the project duration.

We follow the similar procedure to find the other boundary point, X_2. To generate the efficient point with minimum project duration we solve the following problem

$(P_2)'$ $Min\ B$

$Subject\ to\ T = T_{MIN}$

$c1, c2, c3\ and\ c4$

where T_{MIN} is the optimal T value of (P_2). Note $(P_2)'$ finds an optimal B schedule among the smallest T schedules, hence generates an efficient point, having the maximum B value, i.e., B_{MAX}. This efficient point is a boundary point, X_2.

For a sufficiently small e_B, $(P_2)'$ is equivalent to $(P_2)''$

$(P_2)''$ $Min\ T + e_B B$

$Subject\ to\ c1, c2, c3\ and\ c4$

Theorem 6 below, generates a range for e_B that makes $(P_2)'$ and $(P_2)''$ equivalent.

Theorem 6: $(P_2)'$ and $(P_2)''$ are equivalent provided that $e_B < \frac{1}{B_{MAX} - B_{MIN}}$.

Proof: e_B should be small enough so that *T* does not change for any change of *B* value. Since it takes only integer values, the minimum change in *T* value is 1. The maximum change in the *B* value is $B_{MAX} - B_{MIN}$. Hence, $e_B \cdot (B_{MAX} - B_{MIN}) < 1$, i.e., $e_B < \frac{1}{B_{MAX} - B_{MIN}}$ should hold. □

We use $e_B = \frac{1}{B_{MAX} - B_{MIN} + 1}$. Hence, objective function of $(P_2)''$ can be restated as $Min\ T + \frac{1}{B_{MAX} - B_{MIN} + 1} B$

Figure 4.2 illustrates the range for all efficient solutions. The shaded rectangle inside is the range for all efficient solutions. The outer rectangle contains all solutions.

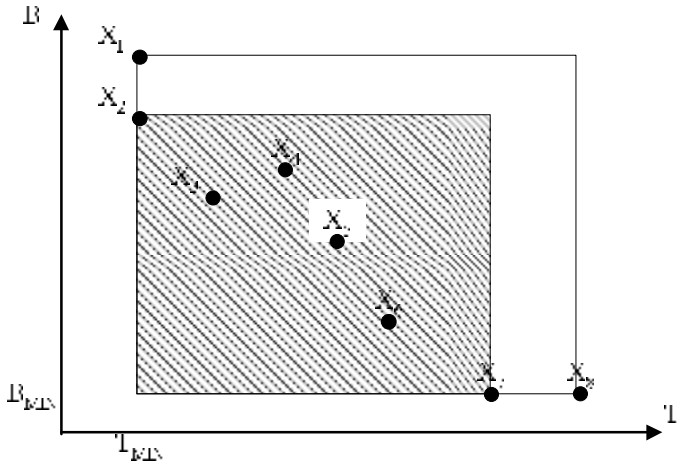

Figure 4.2: Solution Space and Efficient Solutions Space

Constrained Optimization Problem

Consider the following constrained problem, i.e., the Deadline Problem.

(P_t) $Min\ B + e_T T$

$Subject\ to\ T \le t$

$c1, c2, c3\ and\ c4$

An optimal solution to (P_t) is efficient. If we solve (P_t) for all possible value of *t* between (T_{MIN} , T_{MAX}], all efficient solutions are generated. The algorithm below generates all efficient solutions, by varying the *t* value in range (T_{MIN}, T_{MAX}] systematically.

Step 0:

Solve (P_1); (P_2); (P_2)" to find T_{MIN} and T_{MAX} .

Let R=1; $t = T_{MAX}$ -1

Step 1:

Solve (P_t); let the solution be (B^{R+1}, T^{R+1})

Step 2:

R=R+1

If $T^R = T_{MIN}$ ***Then*** Stop, all efficient solutions are generated.

Let $t = T^R$ -1

Go To Step 1

Note that each execution of Step 1 produces an efficient solution. The algorithm is executed *R* times if there are *R* efficient solutions.

Alternatively we can use the following constrained optimization problem to generate all efficient solutions.

(P_b) $Min\ T + e_B B$

$Subject\ to\ B \le b$

$c1, c2, c3\ and\ c4$

An optimal solution to (P_b) is also efficient. We can generate all efficient solutions by varying b in range (B_{MIN} , B_{MAX}], starting by B_{MAX} and terminating when B_{MIN} is reached.

We use to solve Deadline Problem to generate all efficient solutions.

Numerical Example

We illustrate our algorithm, which generates all efficient solutions, on the 5-activity network, depicted in Figure 4.3.

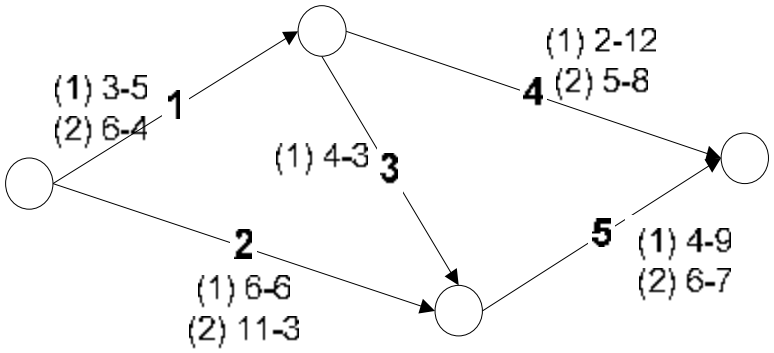

Figure 4.3: AoA Representation of the Sample Project Network

The numbers on each activity indicate the mode of the activity. For example, activity 1 has two modes. The duration of the first mode is 3 and its cost is 5. We have 2, 2, 1, 2, 2 modes for activities 1, 2, 3, 4 and 5 respectively. Therefore, we have a total of $2\times2\times1\times2\times2=16$ feasible solutions. All solutions are enumerated, and determined by the Critical Path Method. The results are reported in Table 4.1, below.

Note that many solutions tabulated are dominated. For example, X_1 is dominated by X_4; X_3, X_6 and X_9 are dominated by X_{12}. The non-dominated, i.e., efficient, solutions are: X_4 =(31, 11); X_{11} =(29, 13); X_{12} =(27, 15); X_{16} =(25, 17). The following figure shows the objective space of the problem. It can be easily observed that all remaining solutions are dominated. The dashed line, called efficient frontier connects the members of the efficient set. It can be easily observed that all solutions above the frontier are dominated.

Table 4.1: All Feasible Solutions of Sample Network

Solution #	Mode					Total Cost	Project Duration
	1	2	3	4	5		
1	1	1	1	1	1	35	11
2	2	1	1	1	1	34	14
3	1	2	1	1	1	32	15
4	1	1	1	2	1	31	11
5	1	1	1	1	2	33	13
6	2	2	1	1	1	31	15
7	2	1	1	2	1	30	14
8	2	1	1	1	2	32	16
9	1	2	1	2	1	28	15
10	1	2	1	1	2	30	17
11	1	1	1	2	2	29	13
12	2	2	1	2	1	27	15
13	2	2	1	1	2	29	17
14	2	1	1	2	2	28	16
15	1	2	1	2	2	26	17
16	2	2	1	2	2	25	17

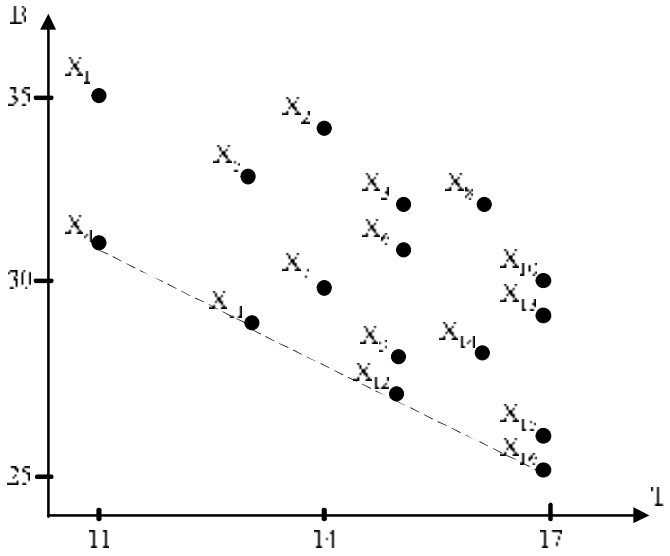

Figure 4.4: Solution Space of the Sample Project Network

Below is the implementation of our algorithm that identifies all four efficient solutions. We first illustrate the generation of these efficient solutions with successive solutions of the Deadline Problem.

Iteration 1

Step 0: Solve (P_1). The optimal solution is (17, 25).

Solve (P_2). The optimal solution is (11, 35), $T_{MIN} = 11$.

(17, 25) is X_{16}, an efficient point r=1. $T_{MAX} = 17$

$t = T_{MAX} - 1 = 16$

$$e_T = \frac{1}{T_{MAX} - T_{MIN} + 1} = 0.142$$

Iteration 2

Step 1: Solve

$Min\ B + e_T T$

$Subject\ to\ T \leq 16$

Let the optimal solution be (T^2, B^2).

(T^2, B^2) =(15, 27) is an efficient point. Note that (15, 27) corresponds to X_{12}.

Step 2: r=2

$T^2 > T_{MIN}$; $t = T^2 - 1 = 14$

Iteration 3

Step 1: Solve

$Min\ B + e_T T$

$Subject\ to\ T \leq 14$

Let the optimal solution be (T^3, B^3).

(T^3, B^3) =(13, 29) is an efficient point. Note that (13, 29) corresponds to X_{11}.

Step 2: r=3

$T^3 > T_{MIN}$

t=T^3-1

Iteration 4

Step 1: Solve

$Min\ B + e_T T$

$Subject\ to\ T \leq 12$

Let the optimal solution be (T^4,B^4).

(T^4,B^4) =(11, 31) is an efficient point. Note that (11, 31) corresponds to X_4.

$T^4 = T_{MIN}$ =11, stop all 4 efficient solutions are generated.

Each execution of Step 1 generates an efficient solution. The algorithm iterates 4 times, hence, there are 4 efficient solutions, i.e., X_{16}, X_{12}, X_{11}, and X_4.

We now illustrate the generation of these efficient solutions with successive solutions of the Budget Problem.

Iteration 1

Step 0: Solve (P_1). The optimal solution is (17, 25), $B_{MIN} = 25$.

Solve (P_2). The optimal solution is (11, 35).

Solve

$Min\ T + e_B B$

$Subject\ to\ B \leq 35$

Let the optimal solution be (T', B')

(T', B') =(11, 31) is an efficient point. Note that (11, 31) corresponds to X_4, $B_{MAX} = 31$.

$b = B' - 1 = 30$

$$e_B = \frac{1}{B_{MAX} - B_{MIN} + 1} = 0.091$$

Iteration 2

Step 1: Solve

$Min\ T + e_B B$

$Subject\ to\ B \leq 30$

Let the optimal solution be (T^2,B^2).

(T^2,B^2) =(13, 29) is an efficient point. Note that (13, 29) is X_{11}.

Step 2: r=2

$B^2 > B_{MIN}$; $b = B^2 - 1 = 28$

Iteration 3

Step 1: Solve

$Min\ T + e_B B$

$Subject\ to\ B \leq 28$

Let the optimal solution be (T^3, B^3).

(T^3, B^3) =(15, 27) is an efficient point. Note (15, 27) is X_{12}.

Step 2: r=3

$B^3 > B_{MIN}$; $b = B^2 - 1 = 26$

Iteration 4

Step 1: Solve

$Min\ T + e_B B$

$Subject\ to\ B \leq 26$

Let the optimal solution be (T^4, B^4).

(T^4, B^4) =(17, 25) is an efficient point. Note that (17, 25) is X_{16}.

Step 2: r=4

$B^4 = B_{MIN}$; stop. All 4 efficient solutions are generated.

CHAPTER 5

COMPUTATIONAL RESULTS

In this section we design an experiment to test the performance of our Branch and Bound Algorithm, heuristics, and to investigate the effects of parameters on the performances of the algorithms. We first discuss our problem instance generation scheme. Later, we present performance measures. Finally we analyze the results of our experiments.

5.1 Data Generation

We use a total of 30 problem instances for the Time/Cost Curve Problem and 80 instances for the Deadline Problem. All instances are taken from Akkan *et al.* (2005b). Small-sized instances are used for the Time/Cost Curve Problem, whereas large-sized instances are used for the Deadline Problem. We next discuss the problem parameters and parameter settings used throughout the experiment.

Problem Parameters

There are two measures used to define the complexity of a project network. These are:

For an AoN representation, CNC is defined as the number of precedence relations divided by the number of activities. Hence a higher CNC results in higher number of arcs, therefore more connected network. For AoA representation, CNC is the ratio of the number of activities to the number of events. Hence increasing CNC results in higher number of activities.

Another parameter used is the number of node duplications needed to transform an AoA network into a series-parallel network. This is referred to as the Complexity Index (CI) and it measures the closeness of the network to a series-parallel one. Each node duplication brings the network closer to the series-parallel network but increases the computational time a multiplicative factor. For the details of the index, the reader is referred to Bein *et al.* (1992). The studies in the literature

point out a relation between the complexity of the solutions and CI, such that increasing CI results in higher solution times. Below, there are the activity dependent problem parameters.

Number of Modes (m): The numbers of modes for all activities are generated from the uniform discrete probability distribution between 1 and 10.

Akkan *et al.* (2005b) generate the time/cost data as follows: The durations are generated from discrete uniform distribution between 3 and 123. Then the durations are sorted such that $t_{i,1} > t_{i,2} > ..... > t_{i,m_i}$. The minimum cost, $c_{i,1}$, is generated from U[5,15] for each activity. Then c_{k+1} is set to $c_k + s_k(t_{i,k} - t_{i,k+1})$ where $s_{k-1} : U[s_k, s_k + 3]$ or $s_{k-1} : U[Max(1, s_k - 3), s_k]$

Number of Activities (N): As CI and CNC define the complexity of the network, they affect the number of activities

Deadline Setting (θ): A set of *t* values are generated for the Deadline Problem, i.e., for constraint $T \le t$. Recall that deadline value *t* should be selected between T_{MIN} and T_{MAX}. Accordingly, *t* values between T_{MIN} and T_{MAX} such that:

$t = T_{MIN} + q\left(T_{MAX} - T_{MIN}\right)$ where θ is referred to as deadline setting.

The Time/Cost Curve Problem is solved for all *t* values between T_{MIN} and T_{MAX}.

Parameter Settings

We solve 30 instances for the Time/Cost Curve Problem using the following parameters; depicted in Table 5.1. CI values are discrete uniform between 4 and 7; 9 and 11 for second and third set of problems respectively. Similarly numbers of activities are discrete uniform between the numbers indicated in Table 5.1 below.

Table 5.1: Time/Cost Curve Problem Parameters

CI	CNC	N	#of instances
0	2	[29,30]	10
[4,7]	2	[34,38]	10
[9,11]	2	[39,42]	10

We solve 80 instances for the Deadline Problem. Each problem instance is solved for 6 different θ values, we set θ=0.15, 0.30, 0.45, 0.60, 0.75, 0.90. Therefore, a total of 480 instances are generated and solved. The parameter combinations are given in Table 5.2.

We set a termination limit of 1 hour for all algorithms. We stop the execution and report the best solution found after 1 hour.

Table 5.2: Deadline Problem Parameters

CI	CNC	N	#of instances
13	5	85	10
13	6	102	10
13	7	[117,119]	10
13	8	[128,129]	10
14	5	85	10
14	6	102	10
14	7	[116,119]	10
14	8	[128,136]	10

5.2 Performance Measures

In this section, we discuss our performance measures used to evaluate the efficiency of our Branch and Bound Algorithm, Heuristics and Lower Bounds.

To evaluate our Branch and Bound Algorithm used for the Deadline Problem we use the following performance measures:

1. CPU time in seconds (average, maximum)
2. Number of nodes generated (average, maximum)
3. Number of unsolved instances, out of 10, in 1 hour.

We use the following performance measures for our heuristics.

1. CPU time in seconds (average, maximum)
2. Percent deviation from the optimal solution (average, maximum)

For lower bound, we only report the percent deviations from the optimal solutions.

To evaluate our Branch and Bound Algorithm for the Time/Cost Curve Problem we use the following performance measures:

1. Solution time in CPU seconds, simply CPU time.
2. Number of efficient solutions
3. CPU time spent per efficient solution (seconds)

The optimal solutions are found by CPLEX 8.1. CPLEX is run for 1 hour and unsolved instances after 1 hour of execution are reported. All experimentations are done in Pentium IV 2.8 GHz, 1GB RAM. All algorithms are coded with MS Visual C++ 6.0.

5.3 Preliminary Experiments

We design a preliminary experiment to evaluate the performance of our mode elimination algorithm (ME), upper bounds (UB), selection (SS) and branching strategies (BS). We define versions of the Branch and Bound Algorithm using different choices of those mechanisms. We now present the results of our preliminary runs.

Branch and Bound Algorithm Versions

We summarize the notation used throughout this section, for different versions of the Branch and Bound Algorithm in Table 5.3.

Table 5.3: Branch and Bound Algorithm Versions

Mode Elimination Algortihm (ME)	
ME_0	No mode elimination
ME_1	Mode elimination only at root node
ME_2	Mode elimination at every node
Upper Bounds (UB)	
UB_1	Improvement 1 at every node
UB_2	Improvement 2 at every node
UB_3	Imp 2 at root node, Imp 1 at every node
Branching Strategy (BS)	
BS_1	Normal Branching Scheme
BS_2	Alternate Branching Scheme
Selection Strategy (SS)	
SS_1	Selection Strategy 1
SS_2	Selection Strategy 2
SS_3	Selection Strategy 3

We use 3 versions for ME, 2 versions for UB, 2 versions for BS and 3 versions for SS. Hence we create a total of 36($3\times3\times2\times2$) combinations of the mechanisms. We evaluate each mechanism separately. We first study the effect of the Mode Elimination Algorithm.

Mode Elimination Algorithm Selection

We run 5 small instances to evaluate performance of the Mode Elimination Algorithms. We select the following problem combination: CI=[0, 14]; CNC=2; N=[31, 44]. Each instance is solved for deadline setting θ=0.15, 0.30, 0.45, 0.60. UB_1, BS_1 and SS_3 are used. Table 5.4 below reports the average CPU times for 5 instances.

Table 5.4: Average CPU Times for Mode Elimination Algorithms

θ	ME_0	ME_1	ME_2
0.15	40.27	39.70	29.61
0.30	13.63	13.37	12.93
0.45	8.39	8.34	8.47
0.60	4.09	4.38	4.15

As can be observed from the table, ME_1 is not significantly different from ME_0. However, ME_2 significantly differs from ME_0 and ME_1, in particular when the deadline constraint gets tighter. This means, if the mode elimination when applied only at root node, does not reduce the CPU time. However, applying mode elimination at each node decreases the average CPU time considerably. Therefore, ME_2 is chosen as the Mode Elimination Algorithm in our main runs.

Branching & Selection Strategy Selection

After ME_2 is chosen as the Mode Elimination Algorithm, we run 4 instances to test the performances of our Branching and Selection Strategies. The instances with CI=[4, 14]; CNC=2; N=[35, 40] are used. Table 5.5 demonstrates the average CPU times and Table 5.6 demonstrates the average nodes evaluated under 4 different deadline settings.

Table 5.5: Average CPU Time (seconds) for Branching and Selection Strategies

θ=0.15	SS_1	SS_2	SS_3	θ=0.30	SS_1	SS_2	SS_3
BS_1	604.20	405.43	10.73	BS_1	115.23	53.69	3.97
BS_2	518.51	344.26	517.62	BS_2	120.59	35.93	11.86
θ=0.45	SS_1	SS_2	SS_3	θ=0.60	SS_1	SS_2	SS_3
BS_1	119.79	78.22	9.28	BS_1	44.37	11.41	4.54
BS_2	135.73	11.96	11.6	BS_2	48.86	5.31	6.69

Moreover large-sized problem instances with CI=13; CNC=5; N=85 are solved. Table 5.7 below shows the average CPU times for each branching, selection strategy over the instances that are solved in 1 hour.

Table 5.6: Average Number of Nodes for Branching and Selection Strategies

θ=0.15	SS_1	SS_2	SS_3	θ=0.30	SS_1	SS_2	SS_3
BS_1	49403	32472	1005	BS_1	7764	3422	259
BS_2	33082	23083	33142	BS_2	5819	1659	580
θ=0.45	SS_1	SS_2	SS_3	θ=0.60	SS_1	SS_2	SS_3
BS_1	8348	5103	587	BS_1	3059	777	303
BS_2	6109	521	495	BS_2	2307	241	305

Table 5.7: Average CPU Times of the Large-Sized Instance

θ=0.15	SS_1	SS_2	SS_3	θ=0.30	SS_1	SS_2	SS_3
BS_1		1167.30	493.67	BS_1		184.92	124.88
BS_2			3331.83	BS_2		326.25	794.80
θ=0.45	SS_1	SS_2	SS_3	θ=0.60	SS_1	SS_2	SS_3
BS_1	9.95	7.00	6.03	BS_1	6.55	3.31	3.09
BS_2	13.82	10.08	10.24	BS_2	9.93	5.42	5.53

We did not include SS_1 when θ=0.15 and θ=0.30 and SS_2 when BS_2 is used and θ=0.15, as all instances remained unsolved after 1 hour.

As can be observed from the tables, the best alternative is BS_1-SS_3 combination. The combination produces the smallest average CPU times over all problems. It also creates the smallest average number of nodes for the majority of the problems.

Recall that BS_1 creates two children for each parent whereas BS_2 creates three children. In absence of any other mechanism, BS_1 is expected to perform better and we observe it is better even we use bounds.

SS_3 outperforms the other strategies in all problem combinations. Recall that SS_3, unlike the other strategies, considers the objective function value, i.e., cost, and branches to the modes with highest difference between next mode. Hence one can expect a better performance from SS_3.

Considering the results of our preliminary runs we conduct our main experiments by the most powerful combination, i.e., BS_1-SS_3.

Upper Bound Selection for an Initial Feasible Solution

We run 7 small instances to compare the effect of our upper bound algorithms on the performance of the Branch and Bound Algorithm. We use small-sized instances with parameters: CI=[0, 14]; CNC=2; N=[29, 40]. BS_1 and SS_3 are chosen as branching and selection strategies. Tables below show the average CPU times and average number of nodes of our Branch and Bound Algorithm for each upper bound algorithm over all deadline settings.

Table 5.8: Average CPU Times of BAB Using Different Upper Bounds

θ	UB_1	UB_2	UB_3
0.15	51.58	84.72	51.72
0.30	8.66	13.10	8.33
0.45	4.63	5.98	4.64
0.60	3.96	5.26	3.96

Table 5.9: Average Number of Nodes of BAB Using Different Upper Bounds

θ	UB_1	UB_2	UB_3
0.15	2937.83	2119.43	2937.83
0.30	429.83	310.17	412.67
0.45	228.83	174.67	228.33
0.60	197.50	162.00	196.17

UB_2 is the best performing upper bound. This follows, the Branch and Bound Algorithm that uses UB_2 creates fewest number of nodes. However this efficiency is achieved at an expense of increased CPU times, as Improvement Heuristic is applied at all nodes. UB_1 and UB_3 show similar performances, both in terms of the CPU times and number of nodes. For some instances, like θ=0.30, UB_3 performs slightly better and we prefer to use UB_3.

Heuristic Branch and Bound Algorithm 2

From Section 3.3 we observe that Alternate Branching Scheme has a significant drawback in CPU time. Since Heuristic Branch and Bound Algorithm 2 uses Alternate Branching Scheme a preliminary experiment is conducted to evaluate the performance of the heuristic. We use ME_2, UB_3, SS_4 settings and run 4 small and 1 large-sized instances. It is observed that although our exact Branch and Bound Algorithm is able to solve all 5 instances with all deadline settings, Heuristic Branch and Bound Algorithm 2 does not terminate in 1 hour for the large-sized instance for θ=0.15. Moreover, it gives an average percentage deviation of 6.14, for the small-sized instances which is not acceptable when it is compared to other heuristics. The results are provided in APPENDIX B. Consequently, Heuristic Branch and Bound Algorithm 2 is not considered in main computational results.

5.4 Main Experiment

We perform our main experiment using BB_1-SS_3 with mode elimination and. UB_3. We first discuss the results for the Deadline Problem and then for the Time/Cost Curve Problem.

5.4.1 Deadline Problem

We first study the performance of the lower bounds. The lower bounds are the objective function values of the optimal LP Relaxed Problem obtained after mode eliminations.

In Table 5.10**,** we report on the average and maximum deviations of the lower bounds from the optimal solution. We compute the deviation for an instance as $\%dev = \frac{OPT - LB}{OPT} \times 100$ where

$OPT =$ Optimal total cost

$LB =$ Optimal LP_relaxed cost

As can be observed from the table, the lower bounds behave consistently well over all problem combinations. All average deviations are below 10% and almost all maximum deviations are below 15%. The satisfactory behavior of the lower bounds can be attributed to the result of Theorem 3. Recall that, the LP Relaxation produces very few, at most two, fractional variables for each activity. Hence the solution is close to the exact solution with no fractional variables. It can also be observed from the table that, the tightness of the deadline affects the power of the lower bounds. As θ increases, the deadlines become larger, hence the deadline constraint gets looser. When the constraint is looser, there are many activities that complete at their longest durations, in the LP Relaxation solution. Hence, the lower bounds perform superior, when θ is bigger. Moreover, when CNC increases, the number of activities increases, and the performance of the lower bounds slightly deteriorate.

We now discuss the performance of our Branch and Bound Algorithm that solves the Deadline Problem exactly. In Table 5.11, we report the average and maximum CPU times and the number of nodes. We assume each unsolved instance contributes to the total CPU time by 3600 seconds, hence those instances are considered while computing average CPU times. We report the results for two values of CI; these values are 13 and 14. The CNC values are between 5 and 8, in unit increments. Recall that, CNC is the ratio of number of arcs to the number of nodes, in the AoA representations. As the arcs represent activities, CNC value directly affects the number of activities, hence the complexity of the problem.

Table 5.10: Lower Bound Deviations

CI	CNC	N	θ	Avg. Dev (%)	Max. Dev. (%)
13	5	85	0.15	7.58	14.92
			0.30	5.04	9.60
			0.45	4.32	9.76
			0.60	2.86	7.53
			0.75	2.48	4.63
			0.90	3.31	11.24
13	6	102	0.15	9.15	13.04
			0.30	7.09	11.20
			0.45	6.06	10.13
			0.60	2.88	5.51
			0.75	2.12	5.90
			0.90	3.48	12.83
13	7	[117,119]	0.15	10.06	14.68
			0.30	9.23	16.92
			0.45	6.92	13.66
			0.60	5.90	12.57
			0.75	4.64	9.46
			0.90	4.41	11.40
13	8	[128,129]	0.15	9.90	16.61
			0.30	7.52	13.05
			0.45	5.81	10.67
			0.60	3.65	9.00
			0.75	3.19	7.78
			0.90	1.43	3.87
14	5	85	0.15	5.52	10.39
			0.30	3.52	7.62
			0.45	3.10	4.93
			0.60	2.85	6.03
			0.75	2.73	6.69
			0.90	2.07	5.62
14	6	102	0.15	6.32	8.51
			0.30	4.27	6.66
			0.45	2.84	4.95
			0.60	1.90	3.54
			0.75	2.29	6.58
			0.90	1.76	5.71
14	7	[116,119]	0.15	9.13	17.66
			0.30	6.73	12.16
			0.45	3.86	8.51
			0.60	3.31	8.49
			0.75	2.14	5.56
			0.90	1.65	4.76
14	8	[128,136]	0.15	9.08	14.12
			0.30	6.93	13.86
			0.45	5.14	10.37
			0.60	4.24	9.25
			0.75	4.17	16.69
			0.90	1.60	6.10

As can be observed from Table 5.11, when CNC increases, the CPU times and in turn number of nodes increase. The increase is more pronounced when CNC is increased from 5 to 6. For example, for CI=13 and θ=0.15, the average CPU times are 1109.42 when CNC=5 (N=85) and 2702.21 when CNC=6 (N=102). When CI=14 these averages become 102.35 and 1626.30 seconds. There are some exceptions for which the increase in CNC leads to a decrease in the CPU times. Some of those exceptions can be explained by our termination limit and the dominant contribution of these instances. Such pronounced exception is for CI=13 and CNC=7, 8. Due to the dominant effect of unsolved instances we experience a decrease in CPU times when CNC becomes 8. We could not observe a significant effect of CI on the complexity of the problem.

The deadline setting "θ" has a significant effect on the problem complexity. Recall that, θ is employed in the deadline constraint as follows: $T \le T_{MIN} + q\left(T_{MAX} - T_{MIN}\right)$. As θ approaches to 1, the activities tend to be closer to their maximum duration modes. As θ gets close to 0, the deadline constraint becomes tighter and, the activities tend to their lower duration modes to maintain feasibility. As the activities become more competing for the tight deadline, lower θ values indicate more complex problems.

We observe the significant effect of θ in our experiments as well. As θ becomes smaller, the CPU times, number of nodes, number of unsolved instances all decrease. When $q \ge 0.6$, all problem instances are solved in very small CPU times. For the most pronounced effect of θ, one can make the following observation. When CI=14, CNC=8, θ=0.15, the average CPU time is 2967.47 seconds with 7 unsolved instances. For the same CI and CNC values, but for θ=0.9, the average CPU time becomes 1.03 seconds and the maximum CPU time is 4.91 seconds, i.e., all instances are solved in less than 5 CPU seconds.

We next discuss the performances of our heuristic procedures. We measure the performance of our procedures using the following deviation measure.

$$\%dev = \frac{Heuristic\ Solution - OPT}{OPT} \times 100$$

Table 5.11: Branch and Bound Algorithm Results for Deadline Problem

CI	CNC	N	θ	Avg. CPU Time	Max. CPU Time	Avg. # of Nodes	Max. # of Nodes
13	5	85	0.15	1109.42	2921.73	44451.8	136461
			0.30	109.83	402.92	3750.7	13211
			0.45	42.96	169.20	1456.2	5351
			0.60	2.45	4.30	94.5	151
			0.75	0.81	1.81	34.8	78
			0.90	0.32	1.14	16.6	62
13	6	102	0.15	2702.21	3600.05 (6)	83498.3	126613
			0.30	1961.88	3600.05 (4)	56954.4	108399
			0.45	470.44	1822.83	12931.2	50412
			0.60	9.36	52.25	277.8	1545
			0.75	1.55	3.33	50.7	117
			0.90	0.78	1.59	32.9	74
13	7	[117,119]	0.15	2700.92	3600.06 (7)	70822.1	122900
			0.30	2032.41	3600.05 (3)	47849.1	98572
			0.45	624.79	3600.02 (1)	14470.1	84122
			0.60	22.59	108.52	564.3	2780
			0.75	2.67	5.14	71.5	140
			0.90	0.78	1.89	25.9	63
13	8	[128,129]	0.15	2408.18	3600.06 (6)	57235	101017
			0.30	1046.96	3600.03 (1)	21209	80195
			0.45	152.93	356.63	3035.4	7034
			0.60	28.64	89.97	637.4	2003
			0.75	4.09	10.06	95	251
			0.90	0.45	1.02	12	26
14	5	85	0.15	102.35	246.95	3927.6	11722
			0.30	63.23	157.58	2110.9	5772
			0.45	36.32	239.61	1354.1	9234
			0.60	6.28	20.64	238	757
			0.75	2.91	12.38	127.7	589
			0.90	0.39	1.05	17.1	47
14	6	102	0.15	1626.30	3600.03 (2)	48056.8	108429
			0.30	64.64	156.64	1573.5	3707
			0.45	31.62	166.64	829.8	4391
			0.60	7.21	23.73	210.3	724
			0.75	0.80	1.70	24.4	55
			0.90	0.60	1.45	23.4	61
14	7	[116,119]	0.15	2253.89	3600.06 (5)	59291.5	109965
			0.30	956.94	3600.03 (2)	21802.5	87960
			0.45	40.93	161.89	887.7	3529
			0.60	17.62	80.83	422.5	2025
			0.75	9.60	80.80	271	2316
			0.90	0.51	1.58	15.2	50
14	8	[128,136]	0.15	2967.47	3600.08 (7)	63345.9	97624
			0.30	1789.93	3600.06 (3)	34766.4	72302
			0.45	170.80	494.80	3378.5	10421
			0.60	18.28	50.02	382	1145
			0.75	7.05	38.75	161.2	907
			0.90	1.03	4.91	28.4	141

*The figures in parenthesis indicate number of unsolved instances in 1 hour (out of 10)

Table 5.12, Table 5.13, Table 5.14 and Table 5.15 report the performances of Construction Heuristic, Improvement Heuristic 1, Improvement Heuristic 2, and Heuristic Branch and Bound Algorithm 1, respectively. The tables report the average and maximum deviations and CPU times. We also include the number of times the heuristic procedures find the optimal solution.

As can be observed from Table 5.12, all deviations are very small. Note that the maximum, average CPU time is 0.04 seconds and the worst CPU time overall instances are 0.06 seconds. Hence the Construction Heuristic produces solutions in negligible CPU times consistently. The CPU times increase slightly with an increase in CNC values, in turn the number of activities. Moreover, increasing θ, leads to a decrease in CPU times. This is due to the fact that LP quickly detects many discrete variables that are close to their maximum durations. Note that, the relatively high CPU times are associated to small θ values.

All average deviations are smaller than 15%. However, we observe a maximum deviation of 38.88% at worst. Hence we cannot conclude that the Construction Heuristic performs consistently well over all instances. We observe that θ has significant effect on the deviations. The smaller and larger θ values lead to better performances. This is due to the fact that the LP tends to maximum and minimum duration modes for large and small θ values, respectively. Moreover, we observe that when θ is small there are about 2 instances out of 10 for which the heuristic solutions are optimal. For example when CNC=7, CI=14 and θ=0.90, there are 4 optimal solutions and the average deviation is 2.07%. On the other hand, for the same combination, but for θ=0.15, the average deviation is 8.81% with nonoptimal solutions. Moreover, we observe slight increase in deviations with increases in CNC values. This is due to the increased problem size brought by more activities.

Table 5.12: Construction Heuristic Results

CI	CNC	N	θ	Avg. CPU Time	Max. CPU Time	Avg. Dev(%)	Max. Dev(%)
13	5	85	0.15	0.02	0.03	7.12	13.40
			0.30	0.02	0.03	9.06	16.18
			0.45	0.01	0.02	12.02	23.62
			0.60	0.01	0.02	10.17	25.80
			0.75	0.01	0.02	8.94	16.19
			0.90	0.01	0.02	5.94	30.40 (2)
13	6	102	0.15	0.02	0.03	9.63	14.21
			0.30	0.02	0.03	13.41	22.95
			0.45	0.02	0.03	13.75	21.16
			0.60	0.02	0.03	9.09	17.71
			0.75	0.01	0.02	7.83	15.79
			0.90	0.01	0.02	8.74	25.65 (2)
13	7	[117,119]	0.15	0.03	0.05	9.40	14.55
			0.30	0.02	0.03	12.81	21.98
			0.45	0.02	0.03	14.00	22.03
			0.60	0.02	0.03	13.50	35.01
			0.75	0.02	0.03	18.18	38.88
			0.90	0.02	0.02	8.16	29.96 (2)
13	8	[128,129]	0.15	0.04	0.05	7.40	13.55
			0.30	0.03	0.05	12.13	22.91
			0.45	0.03	0.03	12.76	30.13
			0.60	0.03	0.03	12.05	26.45
			0.75	0.02	0.03	7.54	21.49
			0.90	0.02	0.02	2.02	5.60 (2)
14	5	85	0.15	0.02	0.03	5.08	8.78
			0.30	0.02	0.03	7.25	19.38
			0.45	0.02	0.03	4.54	9.92
			0.60	0.01	0.02	7.80	25.73
			0.75	0.01	0.02	13.13	25.06
			0.90	0.01	0.02	5.07	12.31 (2)
14	6	102	0.15	0.03	0.03	6.46	10.82
			0.30	0.02	0.03	9.89	16.72
			0.45	0.02	0.03	10.70	19.89
			0.60	0.02	0.03	6.58	17.08
			0.75	0.02	0.02	3.45	10.75 (2)
			0.90	0.02	0.02	4.18	9.65 (2)
14	7	[116,119]	0.15	0.03	0.05	8.81	16.02
			0.30	0.03	0.03	12.05	24.84
			0.45	0.02	0.03	7.65	11.02
			0.60	0.02	0.03	5.65	13.48
			0.75	0.02	0.03	5.17	10.81 (1)
			0.90	0.02	0.02	2.07	12.80 (4)
14	8	[128,136]	0.15	0.04	0.06	11.07	19.03
			0.30	0.03	0.03	10.42	18.97
			0.45	0.03	0.03	9.79	15.96
			0.60	0.03	0.03	12.38	22.54
			0.75	0.02	0.03	5.92	14.41 (1)
			0.90	0.02	0.03	2.53	8.59

*The figures in parenthesis indicate the number of times the optimal solution is found.

Table 5.13: Improvement Heuristic 1 Results

CI	CNC	N	θ	Avg. CPU Time	Max. CPU Time	Avg. Dev(%)	Max. Dev(%)
13	5	85	0.15	0.02	0.03	3.49	7.00
			0.30	0.02	0.03	3.33	7.06
			0.45	0.02	0.02	4.37	12.63 (1)
			0.60	0.01	0.02	2.59	8.12
			0.75	0.01	0.02	3.53	10.31 (2)
			0.90	0.01	0.02	2.85	15.45 (4)
13	6	102	0.15	0.03	0.03	4.63	7.12
			0.30	0.02	0.03	5.67	14.64
			0.45	0.02	0.03	4.91	11.99
			0.60	0.02	0.03	4.14	9.15
			0.75	0.02	0.02	2.61	10.35
			0.90	0.01	0.02	2.40	12.79 (2)
13	7	[117,119]	0.15	0.04	0.05	4.40	8.15
			0.30	0.03	0.03	5.10	7.52
			0.45	0.03	0.03	5.80	14.28
			0.60	0.02	0.03	4.52	17.84
			0.75	0.02	0.03	5.50	13.97
			0.90	0.02	0.02	0.76	2.68 (3)
13	8	[128,129]	0.15	0.04	0.05	3.82	6.85
			0.30	0.03	0.05	4.92	9.63 (1)
			0.45	0.03	0.03	6.32	20.86
			0.60	0.02	0.03	5.42	15.37 (1)
			0.75	0.02	0.03	3.76	8.13 (1)
			0.90	0.02	0.02	0.83	2.67 (4)
14	5	85	0.15	0.02	0.03	2.56	5.51
			0.30	0.02	0.03	1.79	3.84
			0.45	0.02	0.02	2.35	5.45 (1)
			0.60	0.01	0.02	2.54	5.07 (1)
			0.75	0.01	0.02	6.15	19.27
			0.90	0.01	0.02	3.54	12.31 (2)
14	6	102	0.15	0.03	0.03	3.62	6.67
			0.30	0.02	0.03	4.99	9.68
			0.45	0.02	0.03	2.86	6.67
			0.60	0.02	0.03	2.34	7.45 (1)
			0.75	0.02	0.02	1.46	9.93 (3)
			0.90	0.01	0.02	2.70	8.09 (3)
14	7	[116,119]	0.15	0.03	0.05	3.58	7.18
			0.30	0.03	0.03	3.40	9.80
			0.45	0.03	0.03	2.24	5.76 (1)
			0.60	0.02	0.03	3.04	7.65
			0.75	0.02	0.03	2.09	5.65 (1)
			0.90	0.02	0.02	0.90	5.80 (4)
14	8	[128,136]	0.15	0.04	0.05	5.67	15.81
			0.30	0.04	0.05	4.44	15.99
			0.45	0.03	0.03	4.64	10.75 (1)
			0.60	0.03	0.03	3.52	7.21
			0.75	0.02	0.03	1.46	5.75 (2)
			0.90	0.02	0.03	0.49	1.51 (4)

*The figures in parenthesis indicate the number of times the optimal solution is found.

Table 5.14: Improvement Heuristic 2 Results

CI	CNC	N	θ	Avg. CPU Time	Max. CPU Time	Avg. Dev(%)	Max. Dev(%)
13	5	85	0.15	0.29	0.64	2.43	4.96
			0.30	0.19	0.39	1.93	5.14
			0.45	0.14	0.25	2.18	6.34 (1)
			0.60	0.06	0.08	1.54	5.17
			0.75	0.03	0.05	3.08	10.31 (3)
			0.90	0.02	0.02	2.13	12.96 (6)
13	6	102	0.15	0.77	1.28	2.85	5.79
			0.30	0.57	1.05	3.19	5.07
			0.45	0.34	0.84	3.08	11.99 (2)
			0.60	0.14	0.30	1.26	5.65 (1)
			0.75	0.09	0.23	1.23	7.02 (3)
			0.90	0.03	0.05	2.37	12.79 (2)
13	7	[117,119]	0.15	1.20	2.89	2.57	6.54
			0.30	0.97	1.95	3.03	5.50
			0.45	0.57	1.56	2.54	5.28
			0.60	0.20	0.36	2.28	7.25 (1)
			0.75	0.10	0.20	4.74	13.43 (1)
			0.90	0.03	0.05	0.26	2.04 (7)
13	8	[128,129]	0.15	1.71	3.38	2.52	5.06
			0.30	0.96	2.08	3.09	7.35 (1)
			0.45	0.52	1.24	1.51	4.17 (1)
			0.60	0.34	0.64	2.53	11.81 (2)
			0.75	0.14	0.31	1.91	5.66 (1)
			0.90	0.04	0.06	0.61	2.67 (6)
14	5	85	0.15	0.33	0.66	0.96	1.93
			0.30	0.17	0.41	1.34	3.47
			0.45	0.13	0.27	0.90	3.34 (2)
			0.60	0.08	0.16	1.25	5.07 (2)
			0.75	0.04	0.08	2.86	7.69
			0.90	0.02	0.03	1.77	4.11 (3)
14	6	102	0.15	0.56	0.92	1.50	4.04
			0.30	0.55	0.77	2.06	6.98 (1)
			0.45	0.34	0.86	2.26	5.89 (1)
			0.60	0.18	0.44	1.17	2.83 (3)
			0.75	0.07	0.16	1.31	9.93 (4)
			0.90	0.03	0.05	2.10	8.09 (5)
14	7	[116,119]	0.15	0.95	3.47	2.21	5.96
			0.30	0.58	1.44	2.77	9.80
			0.45	0.42	1.27	1.42	4.32 (1)
			0.60	0.21	0.44	1.31	6.56
			0.75	0.10	0.27	1.19	5.48 (1)
			0.90	0.03	0.05	0.87	5.80 (5)
14	8	[128,136]	0.15	1.76	3.88	2.87	5.55
			0.30	1.51	2.45	1.55	2.99
			0.45	0.81	1.95	1.38	4.04 (1)
			0.60	0.45	1.11	2.38	6.56
			0.75	0.16	0.38	0.53	1.84 (3)
			0.90	0.05	0.09	0.44	1.51 (4)

*The figures in parenthesis indicate the number of times the optimal solution is found.

Table 5.15: Heuristic Branch and Bound Algorithm 1 Results

CI	CNC	N	θ	Avg. CPU Time	Max. CPU Time	Avg. Dev(%)	Max. Dev(%)
13	5	85	0.15	69.95	182.92	0.94	4.96 (2)
			0.30	14.27	82.75	0.48	1.37 (3)
			0.45	10.85	61.72	1.00	2.59 (3)
			0.60	1.32	3.31	0.51	3.47 (7)
			0.75	0.57	1.23	0.23	2.33 (9)
			0.90	0.26	0.78	0.47	4.69 (9)
13	6	102	0.15	455.61	1423.97	0.32	1.78 (4)
			0.30	221.53	1370.98	1.17	3.00 (2)
			0.45	54.57	244.41	0.47	2.09 (5)
			0.60	4.28	25.83	0.20	1.32 (6)
			0.75	1.25	2.39	0.13	0.59 (7)
			0.90	0.68	1.42	0.09	0.85 (9)
13	7	[117,119]	0.15	143.12	655.89	1.13	6.44 (2)
			0.30	172.61	738.97	0.97	3.26 (4)
			0.45	35.74	89.94	0.93	2.66 (4)
			0.60	6.16	16.27	0.36	2.06 (7)
			0.75	2.01	6.11	0.80	5.04 (6)
			0.90	0.64	1.27	0.00	0.00 (10)
13	8	[128,129]	0.15	352.52	1322.30	0.85	1.50
			0.30	65.33	344.92	1.13	5.37 (2)
			0.45	31.05	61.16	0.25	1.36 (6)
			0.60	14.97	44.91	0.17	1.26 (7)
			0.75	2.58	6.06	0.01	0.07 (9)
			0.90	0.40	0.92	0.00	0.00 (10)
14	5	85	0.15	24.31	91.88	0.50	1.37 (5)
			0.30	20.30	47.20	0.07	0.69 (8)
			0.45	24.67	161.78	0.21	1.44 (6)
			0.60	3.97	11.14	0.00	0.00 (10)
			0.75	2.57	10.25	0.09	0.88 (9)
			0.90	0.37	0.94	0.00	0.00 (10)
14	6	102	0.15	173.84	555.78	0.46	0.98 (2)
			0.30	21.07	54.19	0.72	2.77 (2)
			0.45	18.89	116.64	0.16	0.66 (5)
			0.60	4.67	15.08	0.00	0.00 (10)
			0.75	0.65	1.36	0.00	0.00 (10)
			0.90	0.57	1.33	0.00	0.00 (10)
14	7	[116,119]	0.15	216.66	1103.63	0.79	2.60 (2)
			0.30	80.11	432.52	0.74	4.40 (2)
			0.45	16.46	58.16	0.12	0.62 (7)
			0.60	9.08	40.73	0.08	0.68 (7)
			0.75	5.92	50.77	0.02	0.20 (9)
			0.90	0.47	1.45	0.00	0.00 (10)
14	8	[128,136]	0.15	1012.24	2585.09	1.10	2.19
			0.30	161.98	603.28	0.75	2.41 (2)
			0.45	34.10	91.22	0.55	1.54 (3)
			0.60	9.46	32.44	0.08	0.64 (6)
			0.75	3.85	22.20	0.10	0.63 (7)
			0.90	1.10	4.47	0.10	1.03 (9)

*The figures in parenthesis indicate the number of times the optimal solution is found.

We next study the performance of the Improvement Heuristics. Recall that, the first improvement heuristic takes the Construction Heuristic as a starting step. Table 5.13 reports the results of Improvement Heuristic 1. We used Improvement Heuristic 1 in our Branch and Bound as an initial feasible solution. We observed that using such a heuristic enhances the efficiency significantly. Hence we expect satisfactory behavior from Improvement Heuristic 1, as an approximate solution. Our results in Table 5.13 support our expectations. We observe that the majority of the average deviations are below 5%. The worst, maximum average deviation is 20.86% (which was 38.88% by Construction Heuristic). Note that the average CPU times are very small. Hence the improvements over Construction Heuristic are obtained in negligible CPU times,. as the CPU times by Construction and Improvement Heuristics are almost the same. The improvement over Construction Heuristic can also be verified by the increase in the number of optimal solutions. The Construction Heuristic could find 21 optimal solutions, whereas Improvement Heuristic 1 finds 43 optimal solutions out of 480 problem instances.

The results for the Improvement Heuristic 2 are given in Table 5.14. Improvement Heuristic 2 is implemented over Improvement Heuristic 1. The deviations are significantly smaller for Improvement Heuristics. Almost all average deviations are below 3% and the worst maximum deviation is 13.43%. The number of optimal solutions is increased to 74. Hence about 20% of the problems are optimally solved by Improvement Heuristic 2. But such improvements are achieved at an expense of increased CPU times. The CPU times are relatively high when compared with the other heuristics, however they are still low. Note that, the majority of the average CPU times are below 1.5 seconds, and the maximum CPU time over all instances is 3.88 seconds.

The results of Heuristic Branch and Bound Algorithm 1 are presented in Table 5.15 above. It can be observed that, the heuristic gives the best solutions over all the heuristics mentioned up to now. Heuristic Branch and Bound Algorithm 1 relies on the idea of Theorem 4**.** The Heuristic Branch and Bound Algorithm 1 assigns non critical activities of the LP Relaxation solution to their highest duration modes. Since there is a chance that non-critical activity in LP Relaxation becomes critical, and is assigned to a lower duration mode in the optimal solution, the algorithm does not guarantee the optimality.

It can be observed from Table 5.15 that, in more than half of the instances (284 out of 480), the heuristic finds the optimal solution. For such instances, the noncritical activities of LP Relaxation solution are turned to be noncritical in the optimal solution as well. For the other instances, we observe very small deviations. Almost all average and maximum deviations are below 1% and 5% respectively. We did not use an initial feasible solution to start the Heuristic Branch and Bound Algorithm 1. Note that we could even obtain better performances with initial feasible solutions.

The Heuristic Branch and Bound Algorithm 1 runs in exponential time. The CPU times are significantly smaller than those of Branch and Bound Algorithm; however they are much larger than those of other heuristics. So it can be favored when a satisfactory solution is required in a tolerable time.

The effect of the parameters on the CPU times is close to that of Branch and Bound Algorithm. The CPU times are not as consistent as deviations.

We also use CPLEX Algorithm to solve the Deadline Problem to optimality. We observe satisfactory CPU times, even better than our Branch and Bound Algorithm for the test instances. The average CPU times of the CPLEX Algorithm and our Branch and Bound Algorithm are tabulated in APPENDIX C.

We finally compare the performance of our heuristics with that of Akkan *et al.* (2005a). The average CPU times and % deviations with respect to different CI, θ and CNC values are reported in Table 5.16, below. The average deviations are relative to optimal solutions for our algorithms. Akkan *et al.* (2005a) report the deviations relative to their lower bounds, as the optimal solutions are not available, so they can be interpreted as upper bounds on actual deviations. The software and hardware used in our study and Akkan *et al.*'s (2005a) study are compatible; hence the CPU times are comparable.

As can be observed from Table 5.16, in all problem combinations, our Improvement Heuristics return better average deviations and smaller CPU times than that of Akkan *et al.* (2005a). The Branch and Bound Based Heuristic, finds much better solutions when compared to all heuristics including Akkan *et al.* (2005a)'s however at an expense of higher CPU times.

Table 5.16: Comparison of Heuristics

	Avg. CPU Times (in seconds)			
CI	Imp. Heur 1	Imp. Heur 2	BaB Heur 1	Akkan *et al.* (2005a)
13	0.02	0.56	103.37	5.57
14	0.02	0.56	114.49	5.29
	Avg. % Deviations			
CI	Imp. Heur 1	Imp. Heur 2	BaB Heur 1	Akkan *et al.* (2005a)
13	4.59	2.41	0.68	7.44
14	3.35	1.71	0.40	6.21
	Avg. CPU Times (in seconds)			
θ	Imp. Heur 1	Imp. Heur 2	BaB Heur 1	Akkan *et al.* (2005a)
0.15	0.03	0.95	306.03	7.45
0.30	0.03	0.69	94.65	5.62
0.45	0.02	0.41	28.29	4.68
0.60	0.02	0.21	6.74	3.96
	Avg. % Deviations			
θ	Imp. Heur 1	Imp. Heur 2	BaB Heur 1	Akkan *et al.* (2005a)
0.15	3.97	2.24	0.76	8.47
0.30	4.20	2.37	0.75	7.54
0.45	4.19	1.91	0.46	6.42
0.60	3.51	1.72	0.17	4.85
	Avg. CPU Times (in seconds)			
CNC	Imp. Heur 1	Imp. Heur 2	BaB Heur 1	Akkan *et al.* (2005a)
5	0.02	0.17	21.21	2.98
6	0.02	0.43	119.31	4.67
7	0.03	0.64	84.99	6.24
8	0.03	1.01	210.21	7.81
	Avg. % Deviations			
CNC	Imp. Heur 1	Imp. Heur 2	BaB Heur 1	Akkan *et al.* (2005a)
5	2.88	1.57	0.46	5.59
6	4.14	2.17	0.44	6.77
7	4.01	2.26	0.64	7.38
8	4.85	2.23	0.61	7.54

5.4.2 Time/Cost Curve Problem

We generate and solve 30 problem instances to test the performance of our Branch and Bound Algorithm for the Time/Cost Curve Problem in generating all efficient solutions. The detailed results of all instances are tabulated in APPENDIX D. We report the average and maximum results in Table 5.17. The table reports the average number of efficient solutions, i.e., the average number of problems solved for each problem instance. The table also includes the average and maximum CPU times spent for each efficient solution and for all efficient solutions of an instance. In all problem combinations, we set CNC to 2 and change CI value to generate problems of different sizes.

Table 5.17: Time/Cost Curve Problem CPU Times (in seconds)

				All Efficient Solutions		One Efficient Solution	
CI	CNC	N	Avg. # of eff.	Avg. CPU Time	Max. CPU Time	Avg. CPU Time	Max. CPU Time
0	2	[29,30]	764.10	4673.71	16008.66	5.21	13.82
[4,7]	2	[34,38]	624.70	9436.60	13872.86	15.23	22.26
[9,11]	2	[38,42]	373.50	18069.60	74244.77	41.80	146.73

The number of efficient solutions is affected by the magnitudes of the activity durations and cost figures. We could not observe the effect of the number of the activities in the number of efficient solutions.

It can be observed from the table that the average time of obtaining an efficient solution is affected by the number of activities, i.e., the CI value. When CI is smaller, we solve the Deadline Problem easier; hence obtain an efficient solution quicker. Note that when CI=0, the average time to obtain an efficient solution is 5.21 CPU seconds. When CI is between 9 and 14, the average time for an efficient solution is 32.74 CPU seconds.

The total time of obtaining an efficient set is influenced by the number of efficient solutions and the time of obtaining an efficient solution.

CHAPTER 6

CONCLUSIONS

In this thesis, we consider a Discrete Time/Cost Trade-off Problem. We study two versions of the problem: minimizing total cost subject to a given deadline and generating all efficient solutions relative to the total cost and project duration criteria, i.e., solving Time/Cost Curve Problem. In the literature, there are some studies that tackle with Deadline and Time/Cost Curve Problems. However, none of them could solve problem instances with more than 50 activities, optimally. Hence efficient solution procedures are required to solve large-sized problem instances. Recognizing this fact, we develop some efficient solution procedures for both problems. For the Deadline Problem, we propose a Branch and Bound Algorithm that uses Linear Programming Relaxation based lower bounds. We also develop several heuristic procedures that are based on Linear Programming Relaxation and branch and bound scheme. To generate all efficient solutions, i.e., to solve Time/Cost Curve Problem, we use the Deadline Problem for all possible realizations of the project duration.

To reduce the size of the Deadline Problem, we generate some mode elimination mechanisms. Moreover, we benefit from the optimal Linear Programming solutions to define some properties of the optimal and near optimal solutions.

Our computational results with up to about 150 activities and 10 modes have revealed the satisfactory behavior of our Branch and Bound Algorithm used to solve the Deadline Problem. We observe that the efficiency of the algorithm is affected by the number of activities and the tightness of the deadline value.

Our heuristic procedures generate solutions that deviate from the optimal solutions by no more than ten percent on average. The two-step heuristic procedure, which constructs an initial feasible solution and makes improvements by interchanges, runs in polynomial time. Hence it generates quick solutions. Branch and Bound Based Heuristic procedure generates higher quality solutions, but at an

expense of increased solution times. So, the decision maker should prefer the two-step procedure if the speed of obtaining solutions is more essential. If quality, is more important then Branch and Bound Based solutions should be favored.

We also perform an experiment to test the efficiency of the algorithm to generate all efficient solutions. As the Deadline Problem could be solved optimally with up to 150 activities, we could generate all efficient solutions for 150 activity problems. In our experiments, we see that the number of the efficient solutions is in hundreds for about 40 activity problems; hence we prefer to limit our runs to 40 activities.

We hope our work stimulates some further research work in project scheduling. Some noteworthy of future research directions can be listed as follows:

- Solving the Budget Problem, i.e., minimizing project duration subject to a constraint on total cost.
- Incorporating resource-constraints: We assume there are unlimited resources, however in many practical situations there may be limited resources for which the activities compete.
- Considering the continuous version of the problem: The continuous version of our problem with linear time/cost relation has been solved polynomial time. Further research may consider nonlinear continuous time/cost functions.

REFERENCES

Akkan, C., Drexl, A., and Kimms, A., (2005a), "Generating an acyclic directed graph with a given complexity index by constraint logic programming", Journal of Logic and Algebraic Programming, 62, pp. 1-39.

Akkan, C., Drexl, A., and Kimms, A., (2005b), "Network decomposition-based benchmark results for the discrete time-cost tradeoff problem", European Journal of Operational Research, 165, pp. 339-358.

Bein, W.W., Kamburowski, J., and Stallmann, M.F.M., (1992), "Optimal reduction of two-terminal directed acyclic graphs", SIAM Journal on Computing, 21, pp. 1112–1129.

Butcher, W.S., (1967), "Dynamic programming for project cost-time curves", Journal of the Construction Division, Proceedings of the ASCE 93, pp. 59-73.

Crowston, W.B., and Thompson, G.L., (1967), "Decision CPM: A method for simultaneous planning, scheduling, and control of projects", Operations Research, 15, pp. 407–426.

De, P., Dunne, E.J., Ghosh, J.B., and Wells, C.E., (1995), "The discrete time–cost tradeoff problem revisited", European Journal of Operational Research, 81, pp. 225–238.

De, P., Dunne, E.J., Ghosh, J.B., and Wells, C.E., (1997), "Complexity of the discrete time–cost tradeoff problem for project networks", Operations Research, 45, pp. 302–306.

Deineko, V.G., and Woeginger, G.J., (2001), "Hardness of approximation of the discrete time–cost tradeoff problem", Operations Research Letters, 29, pp. 207–210.

Demeulemeester, E., Herroelen, W., and Elmaghraby, S.E., (1996), “Optimal procedures for the discrete time/cost trade-off problem in project networks”, European Journal of Operational Research, 88, pp. 50–68.

Demeulemeester, E., De Reyck, B., Foubert, B., Herroelen, W., and Vanhoucke, M., (1998), “New computational results on the discrete time/cost trade-off problem in project networks”, Journal of the Operational Research Society, 49, pp. 1153–1163.

Elmaghraby, S.E., (1993), “Resource allocation via dynamic programming in activity networks”, European Journal of Operational Research, 64, pp. 199-215.

Frank,H., Frisch, I.T., Van Slyke, R., and Chou, W.S., (1971), “Optimal design of centralized computer networks”, Networks, 1, pp. 43–57.

Fulkerson, D.R., (1961), “A network flow computation for project cost curves”, Management Science, 7, pp. 167–178.

Kelley, J.E., (1961), “Critical path planning and scheduling: Mathematical basis”, Operations Research, 9, pp. 296–320.

Meyer, W.L., and Shaffer, L.R., (1965), “Expanding CPM for multiform project time-cost curves”, Journal of the Construction Division, Proceedings of the ASCE, pp. 45–65.

Phillips, S., and Dessouky, M.I., (1977), “Solving the project time/cost tradeoff problem using the minimal cut concept”, Management Science, 24, pp. 393–399.

Robinson, D.R., (1975), “A dynamic programming solution to the cost–time tradeoff for CPM”, Management Science, 22, pp. 158–166.

Rothfarb, B., Frank, H., Kleitman, D.M., Steiglitz, K., and Rosenbaum, D.M., (1970), “Optimal design of offshore natural-gas pipeline systems”, Operations Research, 18, pp. 992–1020.

Skutella, M., (1998), “Approximation algorithms for the discrete time–cost tradeoff problem”, Mathematics of Operations Research, 23, pp. 195–203.

APPENDIX A

NUMERICAL EXAMPLE ON HEURISTICS

Table A.1 below shows the precedence relations and modes of activities of a 29-activity sample network.

Table A.1: Data for the Example Instance

Activity	Successors	Modes
1	3, 4	116-6; 69-524; 56-679; 52-722;15-1043
2	10, 11	109-8; 98-78; 80-231; 73-307; 55-492; 32-773
3	5, 6	118-6; 113-81; 69-681; 59-820; 54-883; 3-1486
4	10, 11	98-11; 20-89
5	7, 8	65-9; 47-84; 39-109
6	10, 11	96-5; 88-72; 85-104; 47-462; 7-835
7	9	87-10; 42-230; 30-277; 9-410
8	10, 11	118-15; 94-121; 88-149; 81-164; 74-181; 64-235; 61-254; 20-590; 6-713
9	10, 11	118-10; 103-95; 98-137; 94-181; 74-419; 31-877
10	12, 13	114-8; 94-97; 82-164; 51-380; 44-449; 36-514; 15-748
11	17, 18	98-9; 72-121; 21-325; 9-388
12	14, 15	48-5; 43-25; 31-52
13	17, 18	120-10; 112-167; 110-211; 97-466; 59-1190; 40-1545; 38-1579; 17-1886; 11-1982
14	16	119-12; 109-157; 100-307; 82-610; 78-671; 49-1175; 42-1300; 30-1509; 17-1773; 13-1844
15	17, 18	109-13; 87-339; 74-550; 67-650; 55-818; 54-835; 44-1003; 29-1247; 7-1615
16	17, 18	88-9; 68-104; 44-201; 30-271
17	19, 20	116-15; 44-386; 41-407; 21-526
18	30	114-8; 64-330
19	21, 22	85-6; 78-50; 62-162; 48-275; 19-472
20	30	119-13; 117-83; 94-839; 80-1286; 64-1773; 56-2010; 53-2098; 45-2331; 15-3181; 8-3383
21	23, 24, 25	87-7; 51-228
22	30	118-8; 106-61; 81-226; 43-561; 35-609
23	26	28-13; 25-16
24	27, 28	120-13; 112-55; 82-140; 64-198; 57-229; 32-408
25	30	104-7; 78-176; 8-579
26	27, 28	103-13; 94-78; 90-106; 86-132; 31-513
27	29	96-12; 75-33
28	30	117- 6; 105-166; 71-608; 58-816; 7-1554; 5-1577
29	30	115-10; 112-41; 85-321; 77-414; 68-496; 66-517; 13-1001

We find that $T_{MAX} = 1503$; $T_{MIN} = 421$ The instance is solved for $q = 0.15$, i.e., $t = T_{MIN} + q\left(T_{MAX} - T_{MIN}\right) = 583$. The fractional variables of LP Relaxation solution are listed as follows:

Table A.2: Fractional Variables of the LP Relaxation

Objective Function Value: 7755.86	
Fractional Variables	
$y_{8,1}$	0.068
$y_{8,5}$	0.931
$y_{9,4}$	0.587
$y_{9,6}$	0.412
$y_{24,5}$	0.960
$y_{24,6}$	0.040
$y_{29,1}$	0.284
$y_{29,7}$	0.715

.

The activity durations of LP solution, i.e., $t_i^{LP} = \sum_{k=1}^{m_i} t_{ik} \times y_{ik}$ are computed below.

$$t_8^{LP} = 0.068 \times 118 + 0.931 \times 74 = 77$$

$$t_9^{LP} = 0.587 \times 94 + 0.412 \times 31 = 68$$

$$t_{24}^{LP} = 0.96 \times 57 + 0.04 \times 32 = 56$$

$$t_{29}^{LP} = 0.284 \times 115 + 0.715 \times 13 = 42$$

The longest modes that are no shorter than t_i^{LP} are listed below,

$$t_{8,5} = 74 < t_8^{LP} = 77; y_{8,5} = 1$$
$$t_{9,6} = 31 < t_9^{LP} = 68; y_{9,6} = 1$$
$$t_{24,6} = 32 < t_{24}^{LP} = 56; y_{24,6} = 1$$
$$t_{29,7} = 13 < t_{29}^{LP} = 42; y_{29,7} = 1$$

The upper bound is found as follows:

$$UB_1 = \sum_{k=1}^{m_i} c_{ik} \times y_{ik} = 8629$$

Improvement Heuristic 1 finds the improvement amounts for all activities as

$IA_1 = c_{1,k} - c_{i,k-1}$ for k's such that $y_{i,k} = 1$

Table A.3 reports IA_i values for all i.

Table A.3: Initial Improvements in Improvement Heuristic 1

Activity	Mode	Improvement Amount
1	5	321
2	1	0
3	6	603
4	1	0
5	3	25
6	1	0
7	4	133
8	5	17
9	6	458
10	7	234
11	1	0
12	3	27
13	1	0
14	2	145
15	1	0
16	4	70
17	4	119
18	1	0
19	5	197
20	1	0
21	2	221
22	1	0
23	2	3
24	5	179
25	1	0
26	5	381
27	2	21
28	1	0
29	7	484

All activities with positive improvement amounts are put in set *PI* and the activity with maximum improvement in *PI* is selected; J=3; IA_3=603.

The mode of Activity 3 is decreased from 6 to 5. The project duration is calculated with CPM. $T = 631 > t = 583$. Note that the solution is infeasible. Then mode of Activity 3 is increased to 6 and it is removed from *PI*.

The activity in *PI* with the maximum improvement is selected; J=29; IA_{29} =484.

The mode of Activity 29 is decreased from 7 to 6. Project duration is calculated with CPM. $T = 604 > t = 583$. Solution is infeasible. The mode of Activity 29 is increased to 6. Activity 3 is removed from *PI*.

Below Table A.4 summarizes all iterations.

Table A.4: Summary of the Execution of the Improvement Heuristic 1

Iter.	J	IA_J	T	Imp.	PI
1	3	603	631	-	{1,3,5,7,8,9,10,12,14,16,17,19,21,23,24,26,27,29}
2	29	484	604	-	{1,5,7,8,9,10,12,14,16,17,19,21,23,24,26,27,29}
3	9	458	589	-	{1,5,7,8,9,10,12,14,16,17,19,21,23,24,26,27}
4	26	381	635	-	{1,5,7,8,10,12,14,16,17,19,21,23,24,26,27}
5	1	321	617	-	{1,5,7,8,10,12,14,16,17,19,21,23,24,27}
6	10	234	601	-	{5,7,8,10,12,14,16,17,19,21,23,24,27}
7	21	221	616	-	{5,7,8,12,14,16,17,19,21,23,24,27}
8	19	197	609	-	{5,7,8,12,14,16,17,19,23,24,27}
9	24	179	581	179	{5,7,8,12,14,16,17,23,24,27}
10	14	145	591	-	{5,7,8,12,14,16,17,23,24,27}
11	7	133	581	133	{5,7,8,12,16,17,23,24,27}
12	17	119	601		{5,7,8,12,16,17,23,24,27}
13	16	70	595	-	{5,7,8,12,16,23,24,27}
14	7	47	591	47	{5,7,8,12,23,24,27}
15	7	220	625	-	{5,7,8,12,23,24,27}
16	24	31	588	-	{5,8,12,23,24,27}
17	12	27	593	-	{5,8,12,23,27}
18	5	25	589	-	{5,8,23,27}
19	27	21	581	21	{8,23,27}*
20	8	17	588	-	{8,23}
21	23	3	583	3	{23}*

At each iteration the activity is shown in column J, improvement amount is shown in column IA_J, the project duration after the corresponding mode change is tabulated in column T. The associated solution is feasible, if the improvement amount is reported in column Imp. The Set *PI* is shown in column PI. We observe

the first successful mode change in the 9th iteration where the mode of Activity 24 is decreased from 5 to 4. Note that, $T = 581 < t = 583$ and Activity 24 is not removed from *PI*. On the other hand, on iterations 19 and 21 the activities 27 and 23 are removed from *PI* respectively. From Table A.3 we observe that Activities 27 and 23 are at their 2nd modes, i.e., they can be moved only once. Moreover, observe that Activity 7 is moved 2 times but its third movement in iteration 15 produces an infeasible schedule. When *PI* becomes an empty set we calculate the upper bound value, as

$$UB_2 = \sum_{k=1}^{m_i} c_{ik} \times y_{ik} = 8246$$

Note that the difference $UB_2 - UB_1 = 383$ is the total improvement amount made by the Improvement Heuristic 1.

We then apply Improvement Heuristic 2 to the same problem instance with deadline setting $q = 0.30$, i.e., $t = T_{MIN} + q\left(T_{MAX} - T_{MIN}\right) = 745$. We calculate the improvements for each activity pair in the solution of Improvement Heuristic 1. At the end of the Improvement Heuristic we found out that $UB_2 = 5962$. Since there are too many activity pairs (29×28), we only show improvement calculations for the Activity 8, in Table A.5 below.

Activity 8 is currently assigned to its 2nd mode. From Table A.1 we know that $c_{8,2}$=121 and $c_{8,1}$=15; $c_{8,3}$=149. Activity 1 has 5 modes and it is assigned to its 5th mode. From Table A.1 we know that $c_{1,5}$=1043 and $c_{1,4}$=722. Since Activity 1 is assigned to its last mode, its mode cannot be increased. Imp(8,1) is the improvement obtained when Activity 8 is assigned to its 3rd mode; and Activity 1 is assigned to its 4th mode, i.e., $\text{Imp}(8,1) = (1043 - 722) + (121 - 149) = 293$

Since, Activity 8 and 2 are not on the same path; improvement of Activity 8-2 pair is not calculated. Activity 3 is assigned to its 1st mode. Hence, its mode cannot be decreased any further. From Table A.1 we know that $c_{3,1}$=6 and $c_{3,2}$=81. Thus, $\text{Imp}(3,8) = (121 - 15) + (6 - 81) = 31$

All remaining improvements are calculated similarly. Table A.5 below gives the improvement amounts of all pairs with Activity 8. The mode column indicates the current mode of the corresponding activity. Note that all activities which are currently assigned to their first modes cannot yield to an improvement. Moreover,

improvements for activities 2, 4, 6, 7 and 9 are not calculated since they are not located on the same path with Activity 8.

After all improvements are calculated Improvement Heuristic 2 executes like Improvement Heuristic 1. The mode change with the maximum improvement amount is selected. The feasibility of the mode change is checked. If mode change is feasible, the mode change is realized. Otherwise, the improvement of the corresponding mode change is set to zero. Algorithm terminates when there are not any positive improvement mode changes.

Table A.5: Improvements of Activity 8 at Improvement Heuristic 2

Activity (J)	Mode	Imp(J,8)	Imp(8,J)
1	5	0	293
2	1	-	-
3	1	31	0
4	1	-	-
5	3	0	-3
6	1	-	-
7	4	-	-
8	2	-	-
9	2	-	-
10	7	0	206
11	1	-6	0
12	3	0	-1
13	1	-51	0
14	1	-39	0
15	1	-220	0
16	4	0	42
17	4	0	91
18	1	-216	0
19	5	0	169
20	1	36	0
21	2	0	193
22	1	53	0
23	2	0	-25
24	5	-73	3
25	1	-63	0
26	5	0	353
27	1	85	0
28	1	-54	0
29	7	0	456

Improvement Heuristic 2 makes 202 iterations and makes the single feasible move at the 99th iteration. Mode of the Activity 8 is decreased to 1 and mode of Activity 3 is increased to 2. Move is feasible as $T = 745 \leq t = 745$ and the improvement is 31 as $\mathrm{Imp}(3,8) = 31$. We calculate the upper bound value, as

$$UB_3 = \sum_{k=1}^{m_i} c_{ik} \times y_{ik} = 5931$$

Note that the difference $UB_3 - UB_2$ =31 is the total improvement amount made by the Improvement Heuristic 2

APPENDIX B

PRELIMINARY RUN RESULTS OF HEURISTIC BRANCH AND BOUND ALGORITHM 2

Table B.1: Average CPU Times and Deviations of 4 Small-Sized Instances

		Average CPU Times (in seconds)	
θ	Average % Deviations	Heuristic BAB	Exact BAB
0.15	6.14	5.75	10.73
0.30	2.80	7.14	3.97
0.45	1.85	6.71	9.28
0.60	2.17	2.21	4.54
0.75	0.50	0.55	1.13
0.90	4.34	0.22	0.23

Table B.2: Average CPU Times and Deviations of the Large-Sized Instance

		CPU Times (in seconds)	
θ	% Deviations	Heuristic BAB	Exact BAB
0.15	4.65	3600.04	493.67
0.30	0.69	773.73	124.88
0.45	12.67	678.46	6.03
0.60	0.00	5.17	3.09
0.75	7.75	2.95	0.36
0.90	6.57	0.09	0.39

APPENDIX C

CPLEX AND BRANCH AND BOUND ALGORITHM CPU TIMES

Table C.1: CPLEX and Branch and Bound Algorithm CPU Times (in seconds) for CI=13 Instances

				Cplex		Branch and Bound	
CI	CNC	N	θ	Avg. CPU Time	Max. CPU Time	Avg. CPU Time	Max. CPU Time
13	5	85	0.15	4.70	11.58	1109.42	2921.73
			0.30	0.78	2.22	109.83	402.92
			0.45	0.28	0.92	42.96	169.20
			0.60	0.09	0.16	2.45	4.30
			0.75	0.06	0.09	0.81	1.81
			0.90	0.06	0.08	0.32	1.14
13	6	102	0.15	68.93	351.44	2702.21	3600.05
			0.30	29.08	189.97	1961.88	3600.05
			0.45	1.67	3.31	470.44	1822.83
			0.60	0.16	0.27	9.36	52.25
			0.75	0.10	0.14	1.55	3.33
			0.90	0.08	0.11	0.78	1.59
13	7	[117,119]	0.15	796.42	3601.36	2700.92	3600.06
			0.30	25.08	112.14	2032.41	3600.05
			0.45	1.26	5.16	624.79	3600.02
			0.60	0.27	0.53	22.59	108.52
			0.75	0.12	0.24	2.67	5.14
			0.90	0.10	0.13	0.78	1.89
13	8	[128,129]	0.15	164.78	616.81	2408.18	3600.06
			0.30	3.66	14.20	1046.96	3600.03
			0.45	0.53	0.97	152.93	356.63
			0.60	0.23	0.33	28.64	89.97
			0.75	0.16	0.23	4.09	10.06
			0.90	0.10	0.19	0.45	1.02

Table C.2: CPLEX and Branch and Bound Algorithm CPU Times (in seconds) for CI=14 Instances

				Cplex		Branch and Bound	
CI	CNC	N	θ	Avg. CPU Time	Max. CPU Time	Avg. CPU Time	Max. CPU Time
14	5	85	0.15	0.76	2.22	102.35	246.95
			0.30	0.30	0.89	63.23	157.58
			0.45	0.14	0.33	36.32	239.61
			0.60	0.11	0.24	6.28	20.64
			0.75	0.08	0.16	2.91	12.38
			0.90	0.05	0.08	0.39	1.05
14	6	102	0.15	16.40	59.47	1626.30	3600.03
			0.30	0.56	1.13	64.64	156.64
			0.45	0.29	0.95	31.62	166.64
			0.60	0.11	0.19	7.21	23.73
			0.75	0.08	0.14	0.80	1.70
			0.90	0.07	0.11	0.60	1.45
14	7	[116,119]	0.15	84.25	482.03	2253.89	3600.06
			0.30	2.82	13.45	956.94	3600.03
			0.45	0.30	0.61	40.93	161.89
			0.60	0.18	0.36	17.62	80.83
			0.75	0.11	0.24	9.60	80.80
			0.90	0.09	0.13	0.51	1.58
14	8	[128,136]	0.15	84.13	267.13	2967.47	3600.08
			0.30	5.42	11.30	1789.93	3600.06
			0.45	0.73	2.06	170.80	494.80
			0.60	0.25	0.38	18.28	50.02
			0.75	0.17	0.27	7.05	38.75
			0.90	0.10	0.13	1.03	4.91

APPENDIX D

DETAILED RESULTS OF TIME/COST CURVE PROBLEM

Table D.6: Time/Cost Curve Results

CI	CNC	N	# of efficient solutions	Total time (in seconds)	Average efficient solution time (in seconds)
0	2	29	854	4848.72	5.68
		29	683	3326.16	4.87
		29	1211	11643.19	9.61
		29	500	1904.2	3.81
		30	1158	16008.66	13.82
		30	366	852.73	2.33
		29	751	1428.33	1.9
		30	661	4719.17	7.14
		29	513	898.97	1.75
		29	944	1107.02	1.17
4-7	2	34	599	9455.97	15.79
		34	565	9560.63	16.92
		35	615	8113.95	13.19
		35	505	8165.8	16.17
		36	497	11061.8	22.26
		36	624	7921.75	12.7
		35	965	13872.86	14.38
		38	554	7201.27	13
		37	739	12994.69	17.58
		38	584	6017.34	10.3
9-11	2	38	506	74244.77	146.73
		42	332	3104.23	9.35
		39	302	10918.7	36.15
		40	401	20250	50.5
		42	275	5880.8	21.38
		38	323	3986.39	12.34
		42	374	2949.27	7.89
		38	324	17365.06	53.6
		38	552	38292.97	69.37
		38	346	3703.78	10.7