THE BUDGET CONSTRAINED DISCRETE TIME/COST TRADE-OFF PROBLEM IN PROJECT NETWORKS

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

BY

GÜVENÇ DEĞİRMENCİ

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

AUGUST 2008

Approval of the thesis:

THE BUDGET CONSTRAINED DISCRETE TIME/COST TRADE-OFF PROBLEM IN PROJECT NETWORKS

submitted by **GÜVENÇ DEĞİRMENCİ** in partial fulfillment of the requirements for the degree of **Master of Science in Industrial Engineering Department, Middle East Technical University** by,

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

Prof. Dr. Nur Evin Özdemirel
Head of Department, **Industrial Engineering** _______________

Prof. Dr. Meral Azizoğlu
Supervisor, **Industrial Engineering Dept., METU** _______________

Examining Committee Members:

Prof. Dr. Ömer Kırca _______________
Industrial Engineering Dept., METU

Prof. Dr. Meral Azizoğlu _______________
Industrial Engineering Dept., METU

Prof. Dr. Sencer Yeralan _______________
Agricultural and Biological Engineering Dept.,
University of Florida, USA

Assoc. Prof. Dr. Yasemin Serin _______________
Industrial Engineering Dept. METU

Assist. Prof. Dr. İsmail Serdar Bakal _______________
Industrial Engineering Dept., METU

Date: August 6, 2008

ABSTRACT

THE BUDGET CONSTRAINED DISCRETE TIME/COST TRADE-OFF PROBLEM IN PROJECT NETWORKS

Değirmenci, Güvenç
M.S., Department of Industrial Engineering
Supervisor: Prof. Meral Azizoğlu

August 2008, 115 pages

The time/cost trade-off models in project management aim to compress the project completion time by accelerating the activity durations at an expense of additional resources.

The budget problem in discrete time/cost trade-off scheduling selects the time/cost mode -among the discrete set of specified modes- for each activity so as to minimize the project completion time without exceeding the available budget. There may be alternative modes that solve the budget problem optimally, however each solution may have a different total cost value.

In this study we aim to find the minimum cost solution among the optimal solutions of the budget problem. We analyze the structure of the problem together with its linear programming relaxation and derive some mechanisms for reducing the problem size. We solve the reduced problem by linear programming relaxation and branch and bound based approximation and optimization algorithms. We find that our branch and bound algorithm finds optimal solutions for medium-sized problem instances in reasonable times and the approximation algorithms produce high quality solutions. We also discuss the way our algorithms could be used to construct the time/cost trade-off curve.

Keywords: Project Management, Time/Cost Trade-off, Budget Problem, Minimum Deadline Problem, Branch and Bound Method

ÖZ

PROJE AĞLARINDA BÜTÇE KISITLI KESİKLİ ZAMAN/MALİYET ÖDÜNLEŞİM PROBLEMİ

Değirmenci, Güvenç
Yüksek Lisans, Endüstri Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Meral Azizoğlu

Ağustos 2008, 115 sayfa

Proje yönetiminde zaman/maliyet ödünleşim problemi bazı aktivitelerin tamamlanma sürelerini azaltarak proje tamamlanma süresini azaltmayı amaçlar. Proje tamamlanma süresinin azalması aktivitelere ek kaynak aktarılarak mümkün olur.

Kesikli zaman maliyet ödünleşimi kapsamında bütçe problemi, projeye ayrılan toplam bütçe sınırını aşmadan, her bir aktivite için proje tamamlanma süresini en aza indirebilecek zaman/maliyet alternatifi seçer. Bütçe probleminin birden fazla optimal çözümü olabilir, ancak her optimal çözümde projenin toplam maliyeti farklıdır.

Bu çalışmada proje toplam maliyeti en az olan optimal bütçe problemi çözümleri üzerine çalıştık. Problemin yapısını analiz ederek, problemin boyutlarını küçültebilecek algoritmalar geliştirdik. Doğrusallık kısıtı kaldırılarak ve dal sınır yöntemi kullanılarak küçülen problemleri çözdük. Geliştirdiğimiz dal sınır yönteminin orta büyüklükteki problemleri kabul edilebilir zamanlarda çözebildiğini, yaklaşım algoritmalarının ise optimale yakın sonuçları bulabildiğini gözlemledik. Ayrıca algoritmalarımızın zaman/maliyet ödünleşim eğrilerinin oluşturulmasında kullanılabileceğini tartıştık.

Anahtar Kelimeler: Proje Yönetimi, Zaman/Maliyet Ödünleşimi, Bütçe Problemi, En az Zaman Sınırı Problemi, Dal Sınır Yöntemi

To My Mother and Grandmother

ACKNOWLEDGMENTS

In this thesis I've worked with a great number of people whose contribution to me is inevitable to mention. It is a pleasure for me to express my thanks to them all in my acknowledgement.

First and foremost, I would like to express thanks to Meral Azizoğlu. Besides her precious supervision on my thesis, she is a role model for me as an academician with her way of thinking and discipline. She is more than a thesis advisor for me who provided an insightful view of my life and with her thoughts, advices she has an impact as much as those of my family members.

I gratefully acknowledge Melih Celik whose contributions to the entire work cannot be underestimated. In the long working hours he really encouraged me to manage the burden of my responsibilities in professional life with his willingness to help. He also deserves my appreciation together with Nihan Gormez Karahan and Ibrahim Karahan for shaping up my research with their fruitful opinions. Many thanks go to Sirin Ozturk and Aras Barutcuoğlu for their sympathy. I also thank Balkar Erdoğan for his commendable teaching on C# and his great friendship.

I gratefully thank Omer Kirca, Sencer Yeralan, Yasemin Serin and Ismail Serdar Bakal for their constructive comments on this thesis. I am very grateful that they accepted to be the members of the examining committee.

I would like to thank specially to Sara Başkentli for her persistent confidence on me. I am extraordinary fortunate for having her love and great support.

My parents deserve special mention for their everlasting support and prayers. My mother, Aysel Değirmenci, felt all my distress and without being tired she encouraged me to relieve my stress. Feeling the prayers of my grandmother, Müşerref Soner, help me to become more patient. It is very valuable to know that my brother Evren Değirmenci aids whenever I call for.

Finally I would like to show gratitude everybody who was important to the successful realization of the thesis, as well as expressing my apology that I could not mention personally one by one.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Project management involves the planning and organizing activities and resources so as to create a desired product or service. During the planning phase of a project, project management must consider many concerns as well as those involving time, cost (money resource) and physical resource aspects. Classical network planning techniques like Critical Path Method and Program Evaluation and Review Technique, essentially study the time aspect. These methods aim to minimize the project duration, assuming that all resources required by the project are available. However, in practice, the successful project completion requires the use of various resources like money, manpower, materials and equipment.

The limited availability of the project resources adversely affects many factors, including the planning objectives, time estimations and project scheduling. Project scheduling is concerned with the allocation of the resources and timing of the activities subject to the precedence and resource constraints.

The various resource problems that may appear during the project scheduling phase can be divided into two classes: time/cost trade-off problem and the resource planning problem. The resource leveling and resource allocation are two well recognized problem areas in resource planning. Resource levelling occurs, when sufficient resources are available and one tries to keep the resource usage as much as possible at a constant rate. The resource allocation problem occurs when total resource usage is restricted to a given limit and the objective is to allocate various resources to the activities in such a way that the project completion time is minimized.

The resource planning problems assume that the resource usages cause conflicts in a sense that the activities cannot be started on time due to the unavailability of the resources or the activities requiring the same resource which is only available one unit at a time must be delayed. The time/cost trade-off problems appear when there are no constraints imposed on the unit time availability of the resources and the activities can be performed at different durations according to their resource usages.

In time/cost trade-off models, there are options of accelerating activity durations by putting additional resource funds. These additional funds may be spent through physical resources like machine purchases and labor recruitment or alternate processing options like overtime and subcontracting. The aim is to reduce the project completion time by putting additional resources, hence accelerating some of the activities. The models that tackle with this time/resource trade-off are refered to as time/cost trade-off models in the project management literature.

The discrete version of the time/cost trade-off problem considers limited number of time/cost alternatives, so called modes, for each activity. Such a consideration is important as the discrete alternatives are very common in practice and discretization provides a convenient means of studying a general time/cost structure (see, for example Hindelang and Muth, 1979; Robinson, 1975). The aim is to select a mode for each activity so that the desired project goals are reached, this forms the so called discrete time/cost trade-off problem.

The discrete time/cost trade-off problems have been studied under two main versions: the deadline problem and the budget problem. The deadline problem minimizes the total cost over all mode assignments subject to the constraint that a given deadline on the project completion time is met. The budget problem minimizes the project time without exceeding the budget allocated over mode assignments. There may be several alternative modes that solve the budget (deadline) problem optimally, however each solution may have a different total cost (project completion time) value. A reasonable aim is to find the minimum cost (project completion time) solution among the ones that optimally solve the budget (deadline) problem.

The deadline problem or the budget problem can be used to generate all nondominated solutions with respect to the project completion time and total cost criteria. The problem of generating all nondominated solutions is refered to as the time/ cost curve problem in the project management literature. The time/cost curve can be derived by solving the deadline (budget) problem for all possible realizations of the project completion time (total cost) values.

The deadline problem has been the subject of several research, since the early sixties. Many approximation and optimization algorithms have been proposed and their use in solving the time/cost trade-off problem has been discussed. However, despite its obvious practical importance, we are aware of only three studies on the budget problem. One of the studies shows that the budget problem –as well as the deadline problem—is strongly NP-Hard. The other studies propose dynamic programming algorithms that run into computational troubles with an increase in the problem size. In this thesis, recognizing this important gap in the literature, we study the budget problem. We aim to find the minimum cost solution among the ones that solve the budget problem optimally. We analyze the structure of the problem together with its linear programming relaxation, and derive some mechanisms en route to reducing the problem size. We solve the reduced problem by linear programming and branch and bound based approximation and optimization techniques. We find that our branch and bound algorithm finds optimal solutions for large sized (medium sized) problem instances in reasonable times and the approximation algorithms produce near optimal solutions very quickly for the minimum deadline (budget) problem. We also discuss the way our algorithms could be used to generate the time/cost trade-off curve.

The rest of thesis is organized as follows. In Chapter 2, we define the problem, its environment and present the mathematical programming model. The related literature is also given in this chapter. In Chapter 3, we present the main body of our work on the budget problem: the reduction mechanisms, the linear programming relaxation of the model, the branch and bound algorithm and approximation algorithms together with the bounding approaches for the budget problem. We also present a procedure to generate the time/cost trade-off curve by using the budget problem. Chapter 4 reports the results of our computational experiment. We conclude in Chapter 5 by stating the main results of our work and pointing out some directions for future research.

CHAPTER 2

PROBLEM DEFINITION AND LITERATURE REVIEW

In this chapter, we first give general information on projects and discuss the single-mode and multi-mode project scheduling problems. The multi-mode problems form the so called discrete time/cost trade-off problems. We then give the mathematical formulations of the discrete time/cost trade-off problems. Finally, we review the related literature.

2.1 Project in General

Project Management Institute defines project as a 'temporary endeavor undertaken to create a unique product or service'. A project can be viewed as an interrelated set of tasks or activities that share resources. The resources are usually scarce and they may be either physical, like labor, machine, equipment or simply money.

In project management terminology, terms activity and task are used interchangeably to mean the smallest indivisible work element. The interrelation of the activities defines the precedence relations that originate from sharing the same resources or having technological input-output sequences. For instance, activity B may need the output of activity A, hence it can start only after A is completed.

To define precedence relations between the activities, predecessor and successor activities are used. If the start of activity B requires the completion of activity A, then it is stated that activity A is the predecessor of activity B, and B is the successor of A. If activity B can start immediately after activity A is completed, then activity A is the immediate predecessor of A and B is immediate successor of A. The immediate predecessor sets are sufficient to explain all precedence structure. This is due to the transitivity of precedence relations, i.e., if activity A precedes activity B, and activity B precedes activity C, then activity A precedes activity C.

Pictorial representation tools like graphs, charts are used to represent the relations between the activities. Some events like the activity start and completion time and the completion time of the entire project are usually depicted on these representations. The most commonly used pictorial representation tools are Gantt Charts and Project Networks.

Gantt Charts

Gantt chart is one of the oldest tools used in project scheduling. The activities are represented by horizontal blocks in the timeline that are located according to their start and finish times. The arrows connecting horizontal blocks indicate the precedence relations between the activities. A Gantt chart for a 6-activity project is provided in Figure 2-1 below. In this project activities 1 and 2 immediately precede activity 5, activity 3 is the immediate predecessor of activity 4 and the project duration is 162 days.

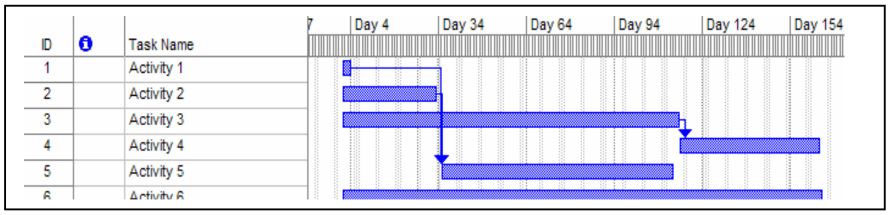

Figure 2-1- A Gantt Chart for the Sample Project

The information that can be depicted from Gantt charts is limited to the start times, finish times and durations of the activities, precedence relations and project completion time. However the critical times for the project, the criticality of the activities cannot be found using these charts.

Project Networks

Project Networks also provide a visual aid for the sequence of the activities. Moreover, they convey information on the critical times of the project and the critical activities. There are two types of project network representations.

- Activity on Arc (AoA) representation
- Activity on Node (AoN) representation

AoA Representation

In AoA representation the activities are represented by arcs, and the events are represented by nodes. The events may be the start and/or completion times of an activity or a set of activities or some particular milestone occurrences, like the half completion of the project or its entire completion. To represent some precedence relations dummy nodes and/or arcs may be needed. The dummy arcs do not consume cost and time. To illustrate the AoA representation, an example taken from Hoare (1973) is used. Table 2-1 shows the immediate precedence relations.

Table 2-1- The Precedence Relations of a Sample Project

Activity	Immediate Predecessors
A	-
B	A
C	A
D	A
E	B
F	C, D
G	D

Figure 2.2 shows the AoA network of the project.

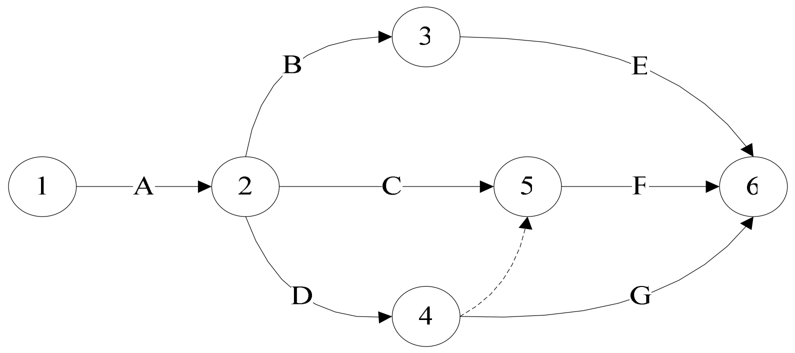

Figure 2-2 The AoA Representation for the Sample Project

In the above network, each node represents an event or a set of events. Node 1 is the source node, indicating the beginning of the project and node 6 is the sink node, indicating the end of the entire project. Node 2 represents not only the end of activity A but also beginning of activities B, C and D. The arc connecting node 1 and node 2 represents activity A. The arc between node 4 and node 5 is a dummy arc for maintaining the precedence relation between activity D and activity F.

AoN Representation

In AoN representations, each activity is represented by a node and each immediate precedence relation is represented by an arc. An arc directed from node A to node B implies that activity A is the immediate predecessor of activity B. The start and end of the project are represented by source and sink nodes, respectively.

Figure 2-3 gives the AoN representation for the sample project.

In Figure 2-3, node *0*, namely source node, represents the start of the project. Activity A is the predecessor of activities B, C and D. Activity F is the successor of activity C and activity D. Node S is the sink node and it represents the end of the entire project.

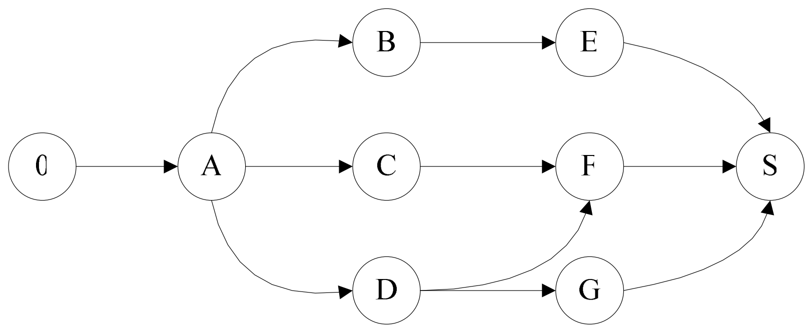

Figure 2-3- The AoN Representation of the Sample Project

2.2 Project Scheduling Problems

Project scheduling involves all decisions regarding the determination of the start and finish times of the activities and the allocation of the scarce resources.

Each activity in a project has a predetermined processing time and a given amount of resource usage. The resource may be labor, machine, tool or available money. Cost is commonly used as an indicator of the resource usage.

According to the number of processing alternatives, two types of projects are defined: single-mode and multi-mode projects. In single-mode projects, there is only one processing alternative for each activity with fixed processing time and cost. On the other hand in multi-mode projects there are more than one alternative for processing time and resource usage. Each alternative, so called mode, has its own processing time (duration) and resource usage (cost).

We now discuss the single-mode and multi-mode project scheduling problems together with their solution algorithms.

2.2.1 Single-Mode Project Scheduling Problems

Single-mode project scheduling problems assign a start time to each activity such that the precedence relations are respected and project is completed in its earliest possible time. The well known Critical Path Method (CPM) is used to find such a schedule. For the sake of completeness, we state the CPM method.

The Critical Path Method (CPM)

CPM finds the earliest and latest start times of the activities, the earliest possible project completion time and defines the critical activities. To state the method we need the following definitions.

Critical Path: The longest path(s) in a project is called critical path.

Critical Activities: Activities that are on the critical path are called critical activities. The earliest and latest start times of the critical activities are equal. Any delay in the start times of the critical activities delays the entire project completion.

Noncritical Activities: Activities that are not on any critical path are noncritical activities.

Total Slack: The difference between the earliest start time and latest completion time of an activity is its total slack. The total slack of an activity is the maximum duration that the activity can use without affecting the earliest project completion time. Accordingly, total slack of a critical activity is equal to its processing time.

In the initialization step, the earliest start times of the activities with no predecessors are set to zero. The earliest start times of the other activities are equal to the maximum of the earliest completion times of their immediate predecessors. After all computations are performed, the earliest project completion time is found. It is equal to the maximum of the earliest completion times of all activities. In order to compute the latest completion times, first, the latest completion times of all activities with no successors are set to the earliest project completion time. Then the latest start times of these activities are found. The latest completion times of other activities are equal to the minimum of the latest start times of their immediate successors. Having known the earliest start and latest times, the total slack values are computed and the critical activities are defined.

Throughout the thesis, we use the following notation:

t_i:	Processing time of activity i.
P_i:	Set of immediate predecessors of activity i.

S_i: Set of immediate successors of activity i.

The followings are returned by the CPM method.

ES_i: Earliest start time of activity i.

LS_i: Latest start time of activity i.

EC_i: Earliest completion time of activity i.

LC_i: Latest completion time of activity i.

Crit: Set of critical activities

$Slack_i$: Total slack of activity i.

Using the notation, we define the algorithm as follows:

```
Initialization:
$ES_i = 0$          $i : P_i = \varnothing$
Main Body:
**Repeat**
        $ES_i = \underset{j \in P_i}{Max}\left\{ES_j + t_j\right\}$          $i : \forall j \in P_i\ ES_j$ is calculated
**Until** $ES_i$ for $i = 1, 2, ...., N$ are calculated
$T = \underset{i}{Max}\left\{ES_i + t_i\right\}$
$LC_i = T$          $i : S_i = \varnothing$
**Repeat**
        $LC_i = \underset{j \in S_i}{Min}\left\{LC_j - t_j\right\}$          $i : \forall j \in S_i\ LC_j$ is calculated
**Until** $LC_i$ for $i = 1, 2, ...., N$ are calculated
Finalization:
$Slack_i = LC_i - ES_i$          $i = 1, 2, ...., N$
$Crit = \left\{i = 1, 2, ..., N \mid Slack_i = t_i\right\}$
```

We illustrate the implementation of the algorithm via a 7-activity project whose data are given in Table 2-2.

Table 2-2 – The Precedence Relations and the Durations of a Sample Project

Activity	Immediate Predecessors	Duration (Week)
A	-	2
B	A	1
C	A	1
D	A	2
E	B	3
F	C, D	4
G	D	2

We now give the stepwise implementation of the CPM method on the sample project.

Initialization:

$P_A = Ø$; $ES_A = 0$

Main Body:

$P_B = P_C = P_D = \{A\}$

$ES_B = ES_C = ES_D = ES_A + t_A = 2$

Activity B precedes activity E, hence $ES_E = ES_B + t_B = 3$

Activity C and activity D are predecessors of activity F,

$ES_F = \max\{ ES_C + t_C; ES_D + t_D \} = \max\{3, 4\} = 4$

Activity D precedes activity G, $ES_G = ES_D + t_D = 4$

Earliest project completion time is $T = \max\{ ES_i + t_i \}= ES_F + t_F = 8$

Having found the earliest project completion time, the latest completion times are computed as follows.

$S_E = S_F = S_G = Ø$, hence $LC_E = LC_F = LC_G = T = 8$

$S_D = \{F, G\}$ thus $LC_D = \min\{LC_F – t_F; LC_G - t_G\} = 4$

Activity F succeeds of activity C and activity E succeeds activity B,

$LC_C = LC_F - t_F = 4$ and $LC_B = LC_E - t_E = 5$

Activities B, C and D are successors of activity A,

$LC_A = \min\{LC_B - t_B; LC_C - t_C; LC_D - t_D\} = LC_D - t_D = 2$

The total slack values are computed by using the earliest start and latest completion times. The earliest start times, latest completion times and total slacks are tabulated below.

Table 2-3 - The CPM Calculations for the Sample Project

Activity	Immediate Predecessors	Processing Time (Week)	Earliest Start Times	Latest Completion Times	Total Slack
A	-	2	0	2	2
B	A	1	2	5	3
C	A	1	2	4	2
D	A	2	2	4	2
E	B	3	3	8	5
F	C; D	4	4	8	4
G	D	2	4	8	4

The critical activities are those that have total slack values equal to their processing times. Accordingly, activities A, D and F are critical and they are on the same critical path. The total completion time of the project is equal to the sum of the completion times of these activities. Those activities that have total slack values greater than their processing times are noncritical. These activities can be delayed by the difference between their total slack values and processing times. For instance, activity B has a total slack of 3 weeks and its processing time is 1 week. It can be delayed for 2 weeks without increasing the project completion time.

All earliest start times and latest completion times are shown on the AoA representation of the network. The numbers in parentheses show the processing times. The earliest start time and latest completion time of each activity are shown in boxes. Critical path is shown with dark arrows.

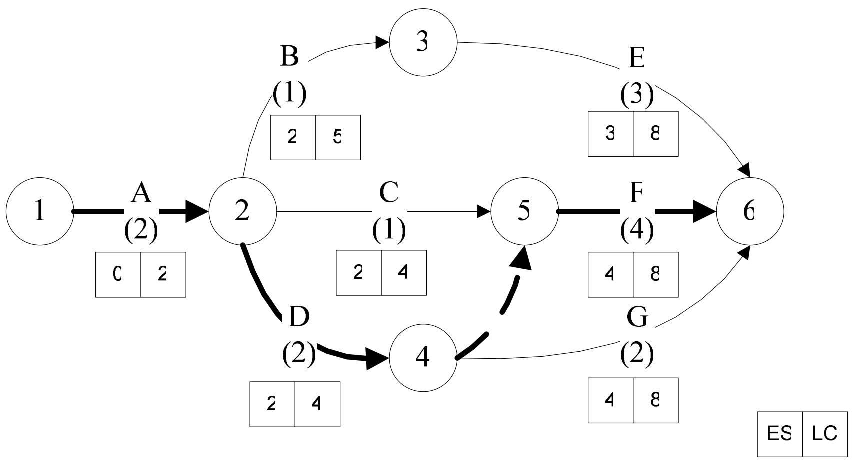

Figure 2-4 - The CPM Computations of the Sample Project

2.2.2 Multi-Mode Project Scheduling Problems

In multi-mode project scheduling problems, there exist several execution modes for at least one activity. Each mode is characterized by its processing time and resource usage and there is usually a trade-off between time and resource usages. An activity can be completed more rapidly if more resources are allocated on. Assuming that the resource usage is reduced to money terms; as the processing time of an activity is shortened, its cost increases. A 5-activity project taken from Battersby (1970) illustrates the time/cost trade-off. The associated information is provided in Table 2-4.

The first activity is the transportation of materials to the construction area. This activity can be completed either in 5 hours or in 4 hours. 3 people can carry the materials in 5 hours; on the other hand a carrying tool is required to complete the activity more quickly. Hence it costs ₤ 40, if the activity is performed in 4 hours, while it costs ₤ 30 if the activity is completed in 5 hours.

Table 2-4 - A Sample Project Illustrating Time/cost Trade-off

Activity	Normal Duration (hr)	Cost £	Rapid Duration (hr)	Cost £
A: Material to site	5	30	4	40
B: Erect hut	6	12	2	20
C: Install electricity	4	10	3	18
D: Install plumbing	5	12	3	20
E: Connect services	3	16	-	-

In the sample project above, there are two modes for each activity except activity E, which has to be completed by a single mode in 3 hours at a cost of £ 16.

The decision in multi-mode problems is to assign a mode for each activity considering total cost and the project completion time criteria. This problem is referred to as the Discrete Time Cost Trade-off (DTCT) problem in the project scheduling literature. The deadline problem, the budget problem and the time/cost curve problems are three different versions of the DTCT problem. The formulations of the deadline and budget problems are provided in the next section.

2.3 Formulations for the Discrete Time/Cost Trade-off Problems

The DTCT problem consists of a set of activities. The activities are represented by index $i \in V = \{0,1,2,...,n+1\}$, where V is the set of activities. Activity 0 and activity $n+1$ are source and sink nodes in the project network, where AoN representation is used.

The precedence relations are defined in set $E \subseteq V \times V$. $\forall (i,i') \in E$ implies that i immediately precedes i' .

For each activity, modes are represented with index $j \in M_i = \{1,...,m_i\}$ and characterized by the following cost and processing time parameters.

c_{ij} : *cost of activity i at mode j*

t_{ij} : *processing time(duration) of activity i at mode j*

According to our convention, if $j < j'$ where $j, j' \in M_i$, then $c_{ij} < c_{ij'}$ and $t_{ij} > t_{ij'}$. In other words 1 denotes the longest and least-cost mode, while m_i denotes shortest and highest-cost mode of activity *i*. For the sample project, for activity A, $c_{A,1} = 30$, $c_{A,2} = 40$ and $m_A = 2$.

Our decision variable that indicates a mode selection for each activity is as defined below.

$$x_{ij} = \begin{cases} 1 & \textit{if activity i is assigned to mode j} \\ 0 & \textit{otherwise} \end{cases}$$

The activity start times are defined as:

S_i *: Start time of activity i*

Accordingly, S_{N+1} denotes the start time of the sink node, therefore the completion time of the entire project. S_{N+1} is called the project length, project duration or project completion time in project scheduling terminology. We use these all these terms interchangeably throughout the thesis.

The constraints of the DTCT problem are defined below.

Each activity should be assigned to exactly one mode. That is,

$$\sum_{j \in M_j} x_{ij} = 1 \qquad \forall i \in V$$

Activity i' cannot start before all its immediate predecessors are completed. That is,

$$S_{i'} \geq S_i + \sum_{j \in M_{i'}} t_{ij} x_{ij} \qquad \forall (i,i') \in E$$

The project starts at time 0, hence all start times are nonnegative. That is,

$$S_0 = 0$$

The partial mode assignments are not allowed

$$x_{ij} \textit{ : binary} \qquad \forall i \in V \textit{ and } \forall j \in M_i$$

The problem contains $\sum_{i=1}^{N} m_i$ binary variables, $N+1$ continuous S_i variables. The number of constraints is equal to $N + \textit{number of elements in set } E$.

The objective of each problem type together with the constraints that are specific to them are presented below.

2.3.1 The Deadline Problem

The objective of the deadline problem is minimizing the total cost of the project which is defined as below.

$$Min \sum_{i \in V} \sum_{j \in M_i} c_{ij} X_{ij}$$

The project should be completed in a predetermined time, say T.

$$S_{N+1} \leq T$$

2.3.2 The Budget Problem

In the budget problem, the total project cost cannot exceed the available project budget, say *b*. Hence the following constraint is required.

$$\sum_{i \in V} \sum_{j \in M_i} c_{ij} X_{ij} \leq b$$

The objective is minimizing the total completion time of the project and is defined as below.

$$Min\, S_{N+1}$$

2.3.3 The Time/Cost Trade-off Curve Problem

The time/cost curve problem involves both project completion time and total cost as criteria. The problem generates all nondominated, i.e., efficient, solutions with respect to two criteria. A solution L is said to be efficient if there does not exist any other solution L' such that $B(L') \leq B(L)$ and $S_{N+1}(L') \leq S_{N+1}(L)$

with strict equality holding at least once, where $B(L)$ and $S_{N+1}(L)$ are the total cost and total project completion time of solution L.

In order to solve the time/cost curve problem, the previous studies solve the deadline problem for all possible realizations of the project completion time. Alternatively, the budget problem for all possible project budget limitations can be used. In this thesis we focus on the budget problem, and in Chapter 3 we discuss how the budget problem can be used to generate time/cost curve.

2.4 Literature Review on the Time/Cost Trade-off Problems

We survey the literature studies on the Time/Cost Trade-off Problem in two parts. The first part is related with the deadline problem and the second one is devoted to the budget problem. The studies in both parts are discussed in chronological order.

2.4.1 The Deadline Problem

Fulkerson (1961) proposes a network flow based solution procedure for the deadline problem with linear time/cost function. The paper is a pioneer work that emphasizes the importance of the linear time/cost trade-off relations. Fulkerson (1961) computes the time/cost curves for all feasible project durations and shows that the associated time/cost curves are convex.

Kelley (1961) also approaches the deadline problem via network flow theory. His network flow algorithm is based on the primal-dual relations. The linear time/cost functions are considered for the activities and the objective is finding the schedule with the maximum project utility. He investigates the structure of the projects and discusses some practical applications of the project utility functions.

Meyer and Schafer (1965) propose one of the earliest approaches for the deadline problem, with discrete time/cost functions. In the study, convex, concave and hybrid time/cost functions are analyzed. They present a mixed-integer programming formulation that does not behave consistent over all problem instances.

Crowston and Thompson (1967) use Decision Critical Path Method (DCPM) that considers the interaction between the scheduling and planning phases of a project. If there exists more than one way to complete a project like performing activity *i* in place of activity *j*, then these activities are shown on the project decision graph. They present a general mixed integer program to select the best project graph and find the associated critical path for the deadline problem. They also propose a heuristic approach. For small projects, the integer program is shown to be manageable. For large projects, the heuristic methods produce promising results.

Crowston (1970) refers to Crowson and Thompson's (1967) study for the Decision Critical Path Method (DCPM) problem. Crowston (1970) proposes repetitive application of the longest path calculations and finds the nondominated set of paths in the decision network. In this way, the DCPM networks are converted to the networks having only the decision nodes with maximum distances in between. Some feasibility and lower bound tests are derived to reduce the network size and the reduced network is solved by a branch and bound algorithm. Crowston (1970) provides some numerical examples to illustrate his procedure, however he does not present a structured computational experiment.

Philips and Dessouky (1977) study the deadline problem with linear time/cost functions. The problem is solved by repetitively applying minimum cuts to the network and reducing the project duration at minimal cost. An application of the procedure is given but no computational experiment is presented.

Hindelang and Muth (1979) propose a Dynamic Programming (DP) algorithm for the deadline problem. The problem formulation is represented by the Decision Critical Path Method (DCPM) model, as in Crowston and Thompson (1967). It is claimed that the shortcomings of the DCPM model is overcome by the proposed procedure. Hindelang and Muth (1979) show that their algorithm can solve the large sized problem instances in reasonable time. However as later shown by De *et al* (1997), their dynamic programming algorithm is flawed.

Elmaghraby (1993) proposes a Dynamic Programming (DP) model for solving the general time/cost trade-off problem. The proposed approach is based on the DP algorithm developed by Fulkerson (1961). First the possible durations are fixed and then further modes are reduced by the DP algorithm. The reduced problem is optimally solved by a branch and bound algorithm. The decision in branch and bound node is the selection of the best mode to the selected activity. Elmagrahraby (1993) does not present any computational results.

De *et al.* (1995) review the previous studies for the Decision Critical Path Method (DCPM) problem together with their drawbacks. They introduce a new dynamic program for the deadline problem and propose that it can be used to construct the time/cost trade-off curve for project networks with no parallel modules. These networks with parallel modules are solved by the network decomposition methods in polynomial time. For the networks with complicated precedence relations, series-parallel conversion is applied.

Demeulemeester *et al.* (1996) address all three versions of the discrete time/cost trade-off problems, i.e. budget, deadline and curve problems. Two dynamic programming based procedures for the network reduction are proposed. The first algorithm intends to find the number of nodes to be reduced in order to convert the project network into a series-parallel network. A series-parallel network consists of networks that can be broken down to series or parallel networks. A series network is a network consisting subnetworks in series, similarly a parallel network is a network consisting subnetworks in parallel. In this study serial optimization is the replacing two arcs in series with a single arc. Similarly two arcs in parallel are replaced for parallel optimization. The second procedure aims minimizing the number of alternative solutions hence increasing the computational efficiency. The procedure uses the serial and parallel merge operations presented by Rothfarb *et al.* (1970) and Frank *et al* (1971).

Philips (1996) presents a solution procedure that uses the cut search approach proposed by Philips and Dessouky (1977). His aim is to present a practical solution procedure for the deadline problem.

De *et al.* (1997) show that the pseudo-polynomial time Dynamic Programming (DP) approach for the Decision Critical Path Method (DTCTP), which is presented by Hindelang and Muth (1979) is incorrect. In the algorithm the different predecessors of an activity that are not considered may cause different earliest start times. Once corrected by using multiple passes of the DP, the resulting algorithm exhibits worst case exponential complexity. De *et al.*(1997) also show that all three versions of the discrete time/cost trade-off problems are strongly NP-hard. However for some structured networks, the problems are polynomially solvable. Hindelang and Muth's (1979) algorithm can solve the deadline problem for pure parallel, pure series and series-parallel networks.

Skutella (1998) reviews the solution algorithms improved by Philips and Dessouky (1977) for the linear time/cost trade-off problem (projects having activities with affine linear and decreasing cost functions). He proposes new approximation algorithms for both the deadline and budget problems. In the proposed approaches performance ratios are guaranteed to be O(log *l*) where *l* is the ratio of the maximum duration of any activity to its minimum nonzero duration.

Demeulemeester (1998) proposes a branch and bound procedure to solve the deadline problem and construct the time/cost trade-off curve. He adapts Fulkerson's algorithm to find a lower bound. Branching is performed by selecting an activity and partitioning its modes into two sets. The results of the computational study shows that the algorithm can solve instances with up to 30 activities and 5 modes per activity.

Vanhoucke and Debels (2005) work on three different extensions of the deadline problem: (1) time/switch constraints, (2) work continuity constraints and (3) net present value maximization. They reuse the exact solution algorithms in the literature and introduce a meta-heuristic approach for generating near optimal solutions for all extensions. They state that the meta-heuristic yields near-optimal solutions for the time switch constrained problem and the results for the net present value problem are promising.

The purpose of the study by Akkan *et al.* (2005) is to define upper and lower bounds for the deadline problems. They add cuts to the LP relaxation of the deadline model and present mode elimination rules. The cuts are defined considering the predecessor-successor relations in order to improve the LP relaxation based lower bounds. Moreover the column generation based on network decomposition is proposed to obtain lower and upper bounds. Their extensive computational results indicate that all bounds are tight and the heuristic approaches are competitive with the existing heuristics.

Hafızoglu (2007) proposes a branch and bound algorithm to solve the deadline problem. They derive some results that are similar to ours. Like us, the algorithms are LP-based.

2.4.2 The Budget Problem

Butcher (1967) studies the budget problem with continuous time/cost functions. He proposes a dynamic programming approach starting with the first activity. The algorithm executes for all activities and all feasible time allocations. The algorithm provides a pseudo-polynomial time solution for the pure series and pure parallel networks.

Robinson (1975) proposes a Dynamic Programming (DP) algorithm to the resource constrained budget problem with discrete time/cost functions. The problem is the allocation of the resources to the activities, considering the precedence and resource limitations. The DP procedure includes the decomposition of the network according to the precedence relations. Optimal policy is stated as follows. If length of a path, say path p, is greater than the length of another path, then additional resources should be allocated to path p. Robinson (1975) does not present any computational result.

In this thesis, we study the budget problem and propose a branch and bound algorithm and heuristic procedures that are capable of solving medium sized instances in reasonable times. The next chapters provide our algorithms and their computational results.

CHAPTER 3

THE BUDGET PROBLEM

Recall that, the budget problem in discrete time/cost trade-off project scheduling minimizes the total project completion time subject to a given budget value. For this problem, there may be many different ways of selecting the activity durations so that the completion times of the resulting schedules are all equal. However, each schedule may yield a different total cost value. Our aim is to select a schedule with the minimum total cost among the ones having minimum total project completion time, under the limited budget.

In the next section, we study the budget problem with the modified objective function and present a methodology to construct the time/cost trade-off curve, i.e., generate all efficient solutions. In Section 3.2 we discuss the minimum deadline problem as a special case of the budget problem, and describe our procedures for reducing the problem size and finding lower bounds, and present algorithms to produce optimal and high quality approximate solutions. In Section 3.3 we study the general budget problem.

3.1 The Modified Budget Problem

The mathematical model for the budget problem is restated below.

$$
\begin{aligned}
& Min\, S_{N+1} \\
& s.to. \\
& \sum_{i \in V} \sum_{j \in M_i} c_{ij} x_{ij} \le b \\
& \sum_{j \in M_j} x_{ij} = 1 && \forall i \in V \\
& S_{i'} \ge S_i + \sum_{j \in M_{i'}} t_{ij} x_{ij} && \forall (i,i') \in E \\
& S_0 > 0 \\
& x_{ij} : binary && \forall i \in V \text{ and } \forall j \in M_i
\end{aligned}
$$

The notation is provided below.

b :	*available budget*
c_{ij} :	*cost of activity i at mode j*
t_{ij} :	*processing time of activity i at mode j*

The decision variables are given as below.

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

S_i : *Start time of activity i*

The objective of the model is assigning each activity to a mode such that the total completion time of the project is minimized and budget limitation is respected. There may be many alternative optimal solutions for the model. In practice, the aim should be the minimization of total completion time by allocating the least possible money, at most the available budget to the project. However the model may return a solution with *(project completion time, total cost)* values of (S^*_{N+1}, B^1) where $B^1 \le b$, and there may exist a solution (S^*_{N+1}, B^2) such that $B^2 < B^1$. We illustrate this case via the following problem instance.

The AoA representation of the project network is given in Figure 3-1. The numbers in the parentheses are the time and cost pairs of the activity modes. For instance, activity A can be executed in 2 weeks at a cost of 150 YTL or in 6 weeks at a cost of 25 YTL. The available budget is set to 3000 YTL.

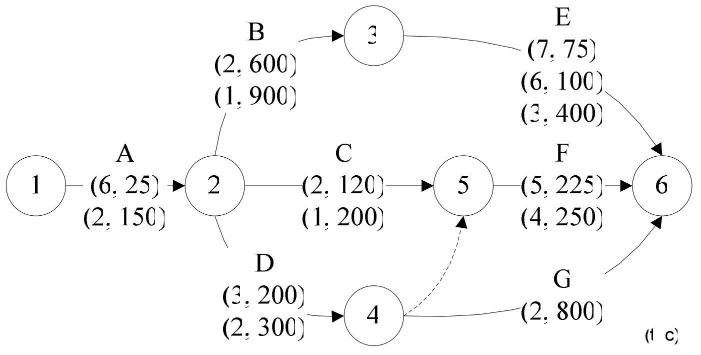

Figure 3-1 - A Sample Multi Mode Project Network

The problem has four alternate optimum solutions for the budget problem. These alternate solutions are depicted in Figure 3-2 through Figure 3-5. In all solutions the assigned modes are shown in bold faces.

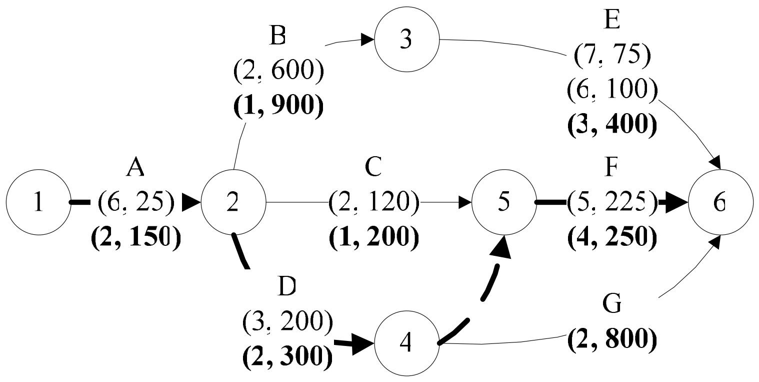

Figure 3-2 – The First Optimal Solution

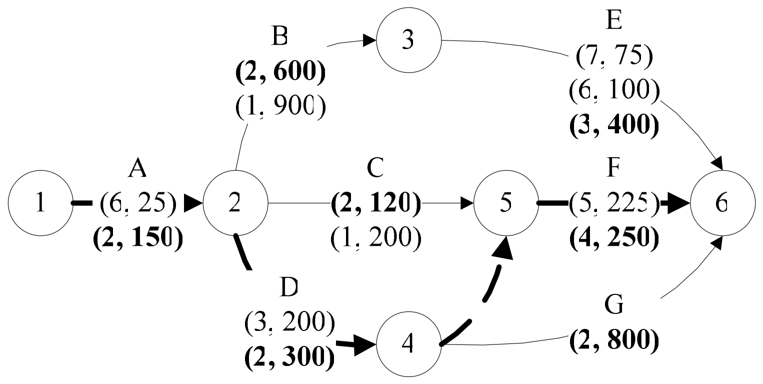

Figure 3-3 – The Second Optimal Solution

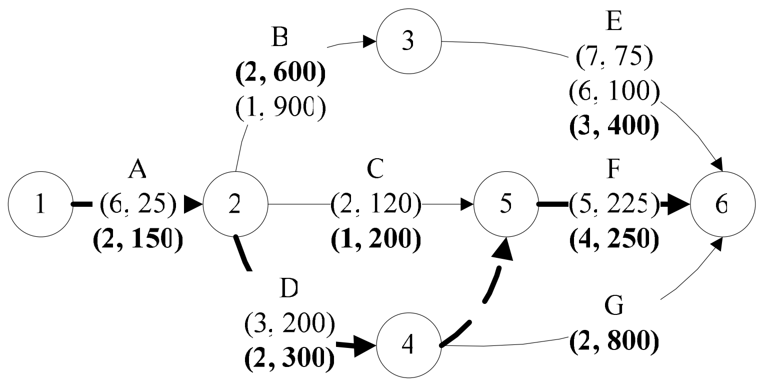

Figure 3-4 – The Third Optimal Solution

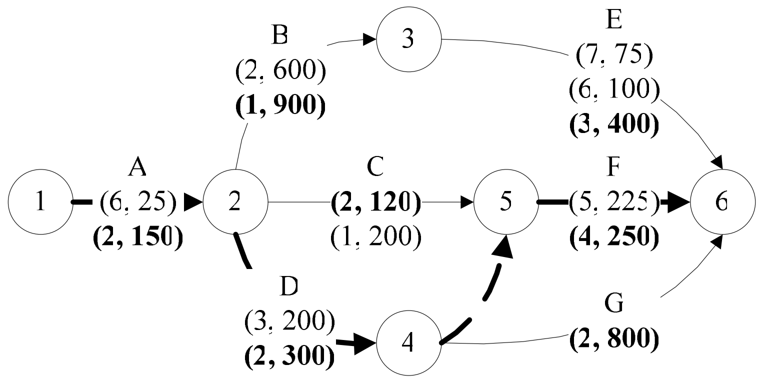

Figure 3-5 – The Fourth Optimal Solution

In all solutions, the S_{N+1} value is at its minimum possible value of 8 weeks.

The first solution has the total cost of 3000 YTL, as found below.

$$\sum_i \sum_j c_{ij} X_{ij} = 150 + 900 + 200 + 300 + 400 + 250 + 800 = 3000$$

The second solution has the total cost of 2620 YTL, as found below.

$$\sum_i \sum_j c_{ij} X_{ij} = 150 + 600 + 120 + 300 + 400 + 250 + 800 = 2620$$

The third solution has the total cost of 2700 YTL, as found below.

$$\sum_i \sum_j c_{ij} X_{ij} = 150 + 600 + 200 + 300 + 400 + 250 + 800 = 2700$$

The fourth and last solution has the total cost of 2920 YTL, as found below.

$$\sum_i \sum_j c_{ij} X_{ij} = 150 + 900 + 120 + 300 + 400 + 250 + 800 = 2920$$

The second solution is nondominated, hence efficient. The other solutions are inefficient as they are dominated by the second solution. A decision maker, who avoids high total cost, prefers the second solution to the others.

The objective function of the budget model is revised in order to eliminate all dominated solutions, and select the efficient one(s). Referring to the above example, we aim to find the second solution, but not the others. To guarantee this, the total cost of the project is added to the objective function of the budget model after weighed by coefficient $\in$. The revised objective function is given below.

$$Min\, S_{N+1} + \in \sum_{i \in V} \sum_{j \in M_i} c_{ij} x_{ij}$$

We hereafter refer to the budget problem with the modified objective function as the modified budget problem. The modified budget problem finds the minimum cost solution among the ones that minimize the total completion time. For a properly selected value of $\in$, the optimal solution of the modified budget problem gives an efficient solution. We find $\in$ as follows: The minimum change in S_{N+1} value is one, when the task times are assumed to be discrete.

The maximum change in the total project cost value, B, is $b_{max} - b_{min}$, where b_{min} and b_{max} are the minimum and maximum total cost values in the efficient solutions' set, respectively. $\in$ should be small enough so that S_{N+1} does not increase for any decrease of B value. This is guaranteed when $S_{N+1} + \in (B + b_{max} - b_{min}) < S_{N+1} + 1 + \in B$, equivalently $\in (b_{max} - b_{min}) < 1$, i.e., $\in < \frac{1}{b_{max} - b_{min}}$.

To guarantee an efficient solution we solve the modified budget problem with $\in < \frac{1}{b_{max} - b_{min}}$. We use $\in = \frac{1}{b_{max}{}^{U} - b_{min}{}^{L} + 1}$ where $b_{min}{}^{L}$ ($b_{max}{}^{U}$) is a valid lower (upper) bound on the b_{min} (b_{max}) value.

The associated objective function is as stated below.

$$Min\, S_{N+1} + \left(\frac{1}{b_{max}{}^{U} - b_{min}{}^{L} + 1}\right)B$$

We hereafter refer to the modified budget problem as $Min\, S_{N+1} + \in B \;\; s.t. \;\; B \leq b$.

In the next section, we discuss a procedure to generate all efficient solutions by solving the modified budget problem. In Section 3.1.2 we discuss the way we find the extreme efficient solutions, i.e., efficient solutions having total cost values of b_{max} and b_{min}. Finally, in Section 3.1.3 we analyze the modified budget problem for general *b*.

3.1.1 Generating All Efficient Solutions, i.e., Solving The Time/Cost Trade-off Problem

All efficient solutions with respect to the total cost and project completion time criteria, can be generated by solving the modified budget problem for all possible values of *b* between ${b_{min}}^{L}$ and ${b_{max}}^{U}$. The procedure below generates all efficient solutions by varying the *b* value, systematically.

Efficient Solution Generation Procedure:

The solution procedure consists of three steps.

Step 0. Find ${b_{min}}^{L}$ and ${b_{max}}^{U}$

Let $b = {b_{max}}^{U}$, $r = 1$

Step 1. Solve the following modified budget problem

$Min\, S_{N+1} + \in B \;\; s.t. \;\; B \leq b$

with $\in = \dfrac{1}{{b_{max}}^{U} - {b_{min}}^{L} + 1}$

If there is no feasible solution, Stop

Let (S^{*}_{N+1}, B^{*}) be the (S_{N+1}, B) values of the optimal solution. (S^{*}_{N+1}, B^{*}) is the *r*th efficient solution.

r = r+1

Step 2. Stop if $B^{*} = {b_{min}}^{L}$

Let $b = B^{*} - 1$

Go to Step 1.

Each iteration of the procedure generates an efficient solution in *Step 1*. When the procedure terminates, all *r* efficient solutions are reached.

The example below illustrates the execution of the procedure. Figure 3-6 gives the AoA representation of the example project.

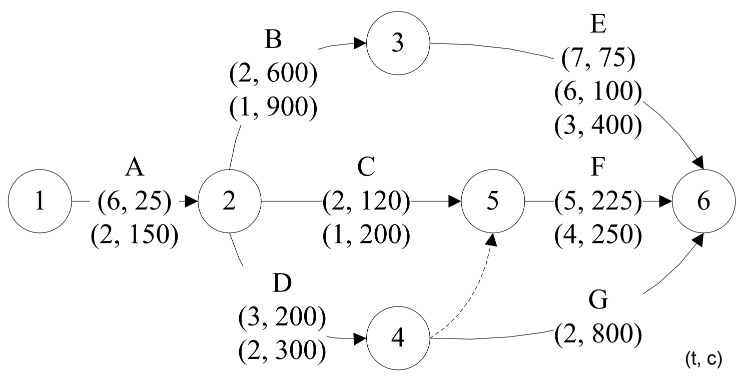

Figure 3-6 - A Sample Multi-Mode Project Network

An upper bound on the b_{max} value of the above project is found as 2620 by using the CPM method. A lower bound on the b_{min} value is computed as 2045 by assigning all activities to their longest duration modes. The efficient solution generation procedure is applied as below.

Step 0. $b_{min}{}^{L} = 2045$ and $b_{max}{}^{U} = 2620$

Let $b = b_{max}{}^{U} = 2620$

Iteration 1

Step 1. The modified budget problem is solved. $(2620,8)$ is the (S_{N+1}, B) values of the optimal solution, and it is the first efficient solution.

Step 2. $B^{*} = 2620 \neq b_{min}{}^{L}$

Let $b = 2620 - 1 = 2619$

Go to Step 1.

Iteration 2

Step 1. The modified budget problem is solved. $(2520,9)$ is the (S_{N+1}, B) values of the optimal solution, and it is the second efficient solution.

Step 2. $B^* = 2520 \neq b_{min}{}^L$

Let $b = 2520 - 1 = 2519$

Go to Step 1.

Iteration 3

Step 1. The modified budget problem is solved. $(2195,10)$ is the (S_{N+1}, B) values of the optimal solution, and it is the third efficient solution.

Step 2. $B^* = 2195 \neq b_{min}{}^L$

Let $b = 2195 - 1 = 2194$

Go to Step 1.

Iteration 4

Step 1. The modified budget problem is solved. $(2170,11)$ is the optimal (S_{N+1}, B) values of the optimal solution, and it is the fourth efficient solution.

Step 2. $B^* = 2170 \neq b_{min}{}^L$

Let $b = 2170 - 1 = 2169$

Go to Step 1.

Iteration 5

Step 1. The modified budget problem is solved. $(2070,14)$ is the optimal (S_{N+1}, B) values of the optimal solution, and it is the fifth efficient solution.

Step 2. $B^* = 2070 \neq b_{min}{}^L$

Let $b = 2070 - 1 = 2069$

Go to Step 1.

Iteration 6

Step 1. The modified budget problem is solved. $(2045,15)$ is the optimal (S_{N+1}, B) values of the optimal solution, and it is the sixth efficient solution.

Step 2. $B^* = 2045 = b_{min}^{\ \ L}$, Stop.

All 6 efficient points are found and shown in the following graph.

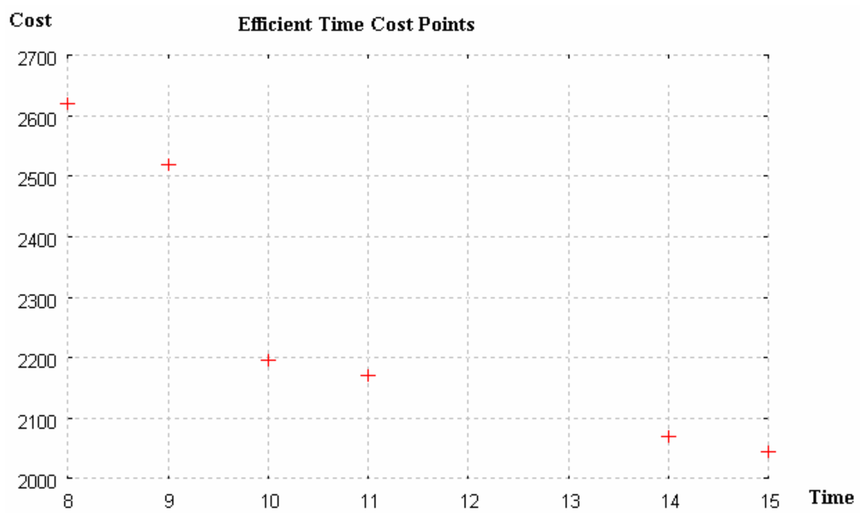

Figure 3-7 – The Efficient Points for The Sample Project

The mode assignments of each efficient solution are tabulated in Table 3-1.

Table 3-1 – The Mode Assignments of Each Efficient Solution

Activity	A	B	C	D	E	F	G
Iteration1 $b = 2620$	2	1	1	2	3	2	1
Iteration2 $b = 2619$	2	1	1	1	3	2	1
Iteration3 $b = 2519$	2	1	1	1	2	1	1
Iteration4 $b = 2194$	2	1	1	1	1	1	1
Iteration5 $b = 2169$	1	1	1	1	2	1	1
Iteration6 $b = 2069$	1	1	1	1	1	1	1

3.1.2 Finding Extreme Efficient Solutions

In this section we present the methods for finding the extreme efficient solutions, i.e., the solutions having total cost values of b_{min} and b_{max}.

Finding the efficient solution with minimum total cost value, b_{min}

For each activity i, the minimum cost is incurred at maximum duration. Hence the minimum total cost will be incurred when all activities are set to their longest duration modes. Formally $b_{min} = \sum_{i \in V} c_{i1}$ where 1 is the first, i.e. least cost (longest duration) mode.

An efficient solution having b_{min} value can be found by the CPM method with the longest activity durations. Let S_{N+1} value of this solution be S_{N+1}^{Max}. (S_{N+1}^{Max}, b_{min}) is an efficient solution, as there cannot exist any other solution having total cost value that is less than b_{min}. Note that S_{N+1}^{Max} is the maximum project completion time value over all efficient solutions.

Finding the efficient solution with maximum total cost value, b_{max}

When all activities are assigned to their minimum duration modes, the CPM method gives the minimum possible project completion time. However, the solution produced by the CPM method may not be efficient, as there may exist another solution having the same project completion time, but smaller total cost value. We let S_{N+1}^{Min} denote the minimum possible project completion time value found by the CPM by setting $t_i = t_{im_i}$ for all i and let $b(S_{N+1}^{Min})$ be the total cost value of the CPM solution. We can find an efficient solution with project completion time value of S_{N+1}^{Min} by solving the following budget problem.

$$(P_1) \quad Min\, S_{N+1} + \in \sum_{i \in V} \sum_{j \in M_i} c_{ij} x_{ij}$$

$$s.to.$$

$$\sum_{i \in V} \sum_{j \in M_i} c_{ij} x_{ij} \leq b(S_{N+1}^{Min})$$

$$\sum_{j \in M_j} x_{ij} = 1 \qquad \forall i \in V$$

$$S_{i'} \geq S_i + \sum_{j \in M_{i'}} t_{ij} x_{ij} \qquad \forall (i,i') \in E$$

$$S_0 > 0$$

$$x_{ij} : binary \qquad \forall i \in V \text{ and } \forall j \in M_i$$

Let (S_{N+1}^{Min}, b_{max}) be the optimum solution to the above problem. b_{max} is the total cost value of an efficient solution having the smallest project completion time and is an upper bound on the total cost values of all efficient solutions.

The solution produced by the above model, has the same critical activities with the $(S_{N+1}^{Min}, b(S_{N+1}^{Min}))$ solution. The critical activities need to be at their shortest possible durations to keep the project completion time at S_{N+1}^{Min}. However the noncritical activities of the CPM solution may not be at their shortest durations, their durations can increase as long as their slack values permit. Note that any increase in the activity duration leads to a reduction in total cost value.

For the sake of simplicity, we hereafter refer to S_{N+1}^{Min} as T_{min}. To illustrate the case, we use a 5-activity network whose AoA representation is given in Figure 3-8.

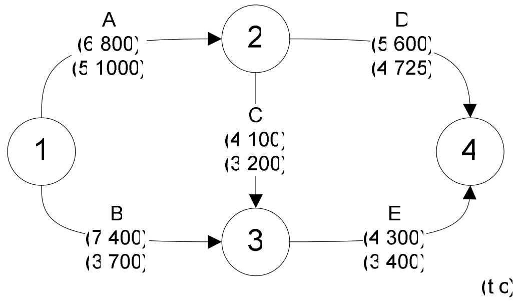

Figure 3-8 - A Sample Network

The numbers in parentheses are (duration, cost) pairs. Note that each activity has two modes.

When all activities are assigned to their shortest duration modes, the critical activities are A, C and E. The project completion time, i.e., T_{min}, is 11 and the total cost is 3025. When all critical activities are assigned to their shortest duration modes and noncritical activities, activity B and D, to their longest duration modes the total completion time of the project does not change but the total cost reduces to 2600. We favor the solution with the smallest total cost, hence the second solution.

We observe that P1 is equivalent to the following deadline problem.

$$
\begin{array}{llll}
(P_2) & Min \sum_{i \in V} \sum_{j \in M_i} c_{ij} x_{ij} & & \\
& s.to. & & \\
& S_{N+1} \leq S_{N+1}^{Min} & & (1) \\
& \sum_{j \in M_j} x_{ij} = 1 & \forall i \in V & (2) \\
& S_{i'} \geq S_i + \sum_{j \in M_{i'}} t_{ij} x_{ij} & \forall (i,i') \in E & (3) \\
& S_0 > 0 & & (4) \\
& x_{ij} : binary & \forall i \in V \; and \; \forall j \in M_i & (5)
\end{array}
$$

Note that as (P_1), (P_2) looks for the minimum total cost solution among the ones that have minimum project completion time. As S_{N+1}^{Min} is the minimum possible completion time, constraint *(1)* will be satisfied as equality. (P_2) is a deadline problem with the minimum possible completion time value and hereafter will be referred to as the minimum deadline problem.

An optimal solution to the minimum deadline problem returns the maximum total cost value of all efficient solutions and defines our b_{max} value.

Rather than solving a budget problem(P_1), we prefer to solve the minimum deadline problem, i.e., (P_2) due to its special structure. This special structure allows us to derive more powerful reduction mechanisms and bounding techniques. Next, we analyze the minimum deadline problem in detail.

3.2 A Special Case – The Minimum Deadline Problem

In this section, first a simple mode fixing algorithm is explained. Second, mode elimination procedures are proposed. Then the optimal Linear Programming Relaxation (LPR) solution and its properties are introduced. The section continues with the lower bounding procedures. Finally, the branch and bound method is presented. Our solution algorithms take their main idea from Hafizoğlu (2007) who deals with the general deadline problem.

3.2.1 Mode Fixing Rule

Recall that in the minimum deadline problem, all critical activities are assigned to their shortest modes in all feasible solutions. To find the critical path, all activities are assigned to their shortest modes, and the Critical Path Method (CPM) is applied. The AoA network in Figure 3-9 is used for discussion purposes.

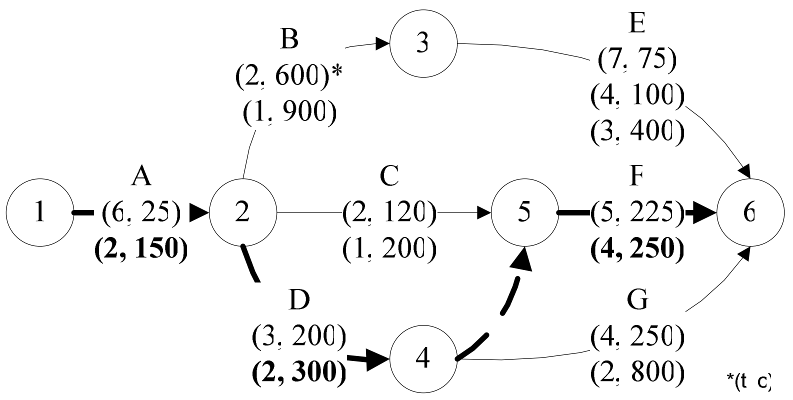

Figure 3-9 - An Example for Mode Fixing

In the above project, activities A, D and F are on the critical path. The length of the critical path is 8. As the critical activities are fixed at their shortest duration modes, the problem is reduced to find the modes for the noncritical activities, B, C, E and G. The optimal mode assignments are given in bold faces in Figure 3-10.

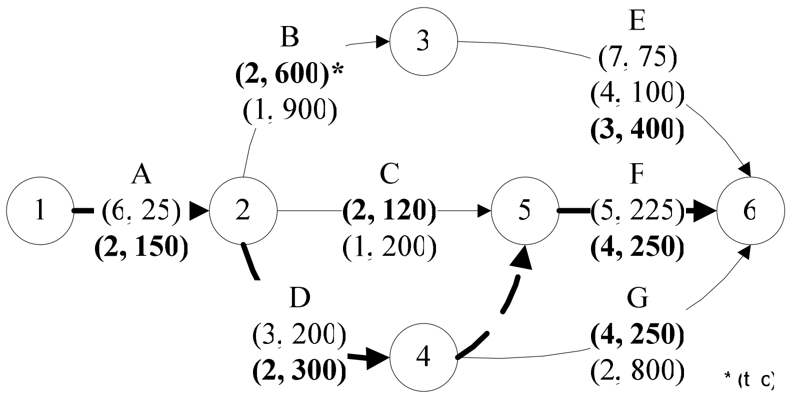

Figure 3-10 - An Optimal Solution for the Example Project

3.2.2 Mode Elimination Rules

In this section, we introduce short mode, long mode and costly mode elimination rules for the noncritical activities. Our aim is to reduce the size of the search by eliminating some modes that are either infeasible or nonpromising, i.e., cannot lead to an optimal solution.

Short Mode Elimination Rules

As the short modes are costly, they are selected only to maintain feasibility. En route to the minimum cost solution, the attempt is to assign the noncritical activities to their longer modes as long as the deadline constraint permits. We need the following notation to discuss the short mode eliminations.

ES_L_i: Earliest start time of activity *i*, when all activities are assigned to their longest modes.

LC_L_i: Latest completion time of activity *i*, when all activities are assigned to their longest modes. LC_L_i is found by the CPM method, using T_{Min} as the deadline.

While computing ES_L_i and LC_L_i we assign all noncritical activities to their longest modes and critical activities to their shortest modes, i.e., we set

$X_{i1} = 1$ where activity i is noncritical

$X_{im_i} = 1$ where activity i is critical

As all critical activities are assigned to their shortest duration modes, ES_L_i, LC_L_i and the following theorem are defined for the noncritical activities.

Theorem 1

If $LC\_L_i - ES\_L_i \geq t_{ij}$ then the modes $j+1$ through m_i for activity i cannot lead to an optimal solution.

Proof: Assume a solution that contradicts with the condition of the theorem, i.e., $LC\_L_i - ES\_L_i \geq t_{ij}$ and $X_{ij'} = 1$ where $j' > j$. Since $c_{ij'} > c_{ij}$ setting $X_{ij} = 1$ in place of $X_{ij'} = 1$ improves the objective function value by $c_{ij'} - c_{ij}$. And such an interchange is feasible as $LC\_L_i - ES\_L_i \geq t_{ij}$. Hence any solution that contradicts with the condition of the theorem cannot be optimal. □

The example whose activity network is given in Figure 3.5 illustrates the application of the rule. The bold face modes show the longest modes used in the CPM method. Using the result of Theorem 1, we consider longer, i.e., less costly, modes as long as total slack values permit.

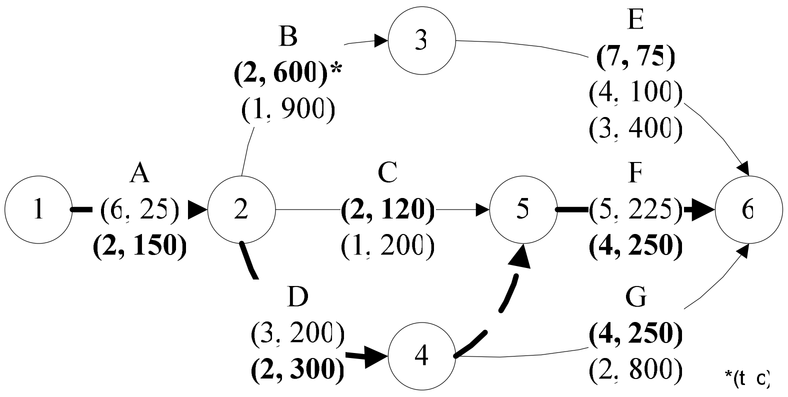

Figure 3-11 – A Sample Network with Noncritical Activities at their Longest Mode

After assigning all noncritical activities, B, C, E and G, to their longest duration modes, ES_L_i and LC_L_i values are computed and tabulated below.

Note that some $LC\_L_i - ES\_L_i$ values are negative. This is due to the fact that assigning some noncritical activities to their longest modes cannot lead to a feasible solution.

Table 3-2 –The Earliest and The Latest Completion Times

Activity	Assigned Duration (Week)	ES_L_i	LC_L_i ($T_{Min} = 8$)	$LC\_L_i - ES\_L_i$
A	2	0	-1	-1
B	2	2	1	-1
C	2	2	4	2
D	2	2	4	2
E	7	4	8	4
F	4	4	8	4
G	4	4	8	4

Theorem 1 eliminates the last mode of activity E as shown below.

Note that $LC\_L_i - ES\_L_i \geq t_{Ej}$ is tested for $j = 1$,

$t_{Ej} = 7$ and $LC\_L_i - ES\_L_i = 4 \not\geq 7$

For j = 2; $t_{Ej} = 4$ and $4 \geq 4$ hence shorter mode, i.e., mode 3, should be eliminated. We drop mode 3, whose processing time is 3.

The idea behind elimination is that the model never assigns activity E to mode 3, as mode 2 would be assigned in a cheaper way even when worst case of the longest mode assignments are assumed. Similarly, for activity C, mode 2 and for activity G mode 2 are eliminated as they would never produce optimal solutions.

Long Mode Elimination

Here long modes that lead to infeasible solutions are eliminated. We use the following notation to state the long term elimination theorem.

ES_E_i: Earliest start time of activity *i*, when all activities are assigned to their shortest modes.

LC_E_i: Latest completion time of activity *i*, when all activities are assigned to their shortest modes.

In computing the ES_E_i and LC_E_i values, the CPM method is used by setting all noncritical activities to their shortest modes and the deadline value to T_{Min}.

$X_{im_i} = 1$ where activity i is noncritical

$X_{i1} = 1$ where activity i is critical

Theorem 2 resembles the one presented in Akkan et. al. (2005) for the general problem.

Theorem 2

$LC\_E_i - ES\_E_i < t_{ij}$ for activity *i* then modes *1* through *j* cannot lead to a feasible solution.

Proof: As all activities are set to their shortest processing times, the resulting total slack, $LC\_E_i - ES\_E_i$, is the maximum assignable duration for activity *i*, en route to a feasible solution. This follows, the modes from 1 through *j* cannot lead to a feasible solution, as the resulting processing times are greater than the maximum total slack value. □

Using the result of Theorem 2, we eliminate some long duration modes that would lead to an infeasible solution.

We illustrate the theorem on the AoA network in Figure 3-2. The bold-face numbers give the shortest modes used in the CPM method.

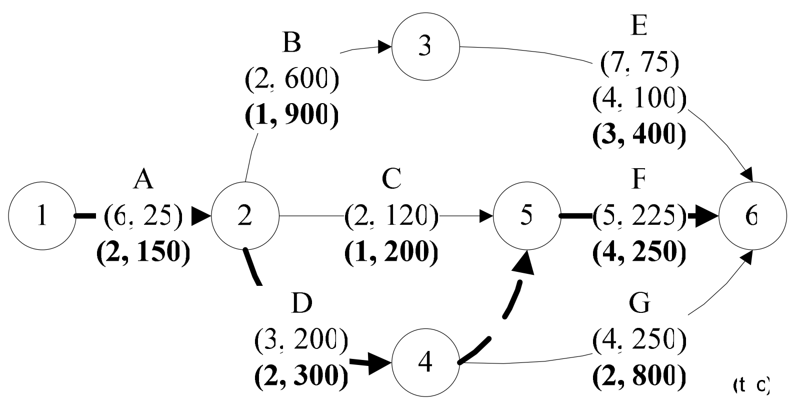

Figure 3-12 - A Sample Network with Noncritical Activities at their Shortest Mode

The associated ES_E_i, LC_E_i and $LC\_E_i - ES\_E_i$ values are tabulated below.

Table 3-3 - The Earliest Start and Latest Completion Times

Activity	Assigned Duration (Week)	ES_E_i	LC_E_i ($T_{Min} = 8$)	$LC\_E_i - ES\_E_i < t_{ij}$
A	2	0	2	2
B	1	2	5	3
C	1	2	4	2
D	2	2	4	2
E	3	3	8	5
F	4	4	8	4
G	2	4	8	4

All noncritical activities (B, C, E and G) are tested whether $LC\_E_i - ES\_E_i < t_{ij}$ or not.

Using the test the longest duration mode of activity E is eliminated as shown below:

$LC\_E_i - ES\_E_i < t_{ij}$ is tested for $j \in M_E$

For j = 1; $t_{Ej} = 7$ and $7 > 5$ hence mode 1 is eliminated.

For j = 2; $t_{Ej} = 4$ and $4 \not> 5$ therefore mode 2 and the other shorter modes cannot be eliminated.

Costly Mode Eliminations

The cost parameters are used to determine nonpromising costly modes. The following notation is used to state the rule.

LB_i : A valid lower bound on the total cost of all activities, other than *i*.

UB : A valid upper bound on the total cost of the project.

Like Theorem 1, Theorem 3 states a rule for eliminating short duration modes.

Theorem 3

If $LB_i + c_{ij} > UB$ then the modes *j* through m_i for activity *i* cannot lead to an optimal solution.

Proof: LB_i is the minimum cost of performing all activities, except activity *i*. Activity *i* cannot be assigned to mode *j* and all other shorter duration modes, in any optimal solution as it cannot lead to a better solution than UB, even when the other activities performed at their lowest possible costs. □

Two different LB_i value can be computed as follows.

Simple Lower Bound

A lower bound on the total cost for the activities except activity *i* is the cost incurred by these activities when they are assigned to their longest modes.

$$NLB_i = \sum_{k \in V, k \neq i} c_{k1}$$

LP Based Lower Bound

LB_i value is computed in two steps.

Step 1. Activity *i* is fixed to its shortest duration mode and its cost is set to zero, i.e, $X_{im_i} = 1$ and $c_{im_i} = 0$.

Step 2. The discrete nature of X_{ij} variables is relaxed and the problem is solved as a continuous LP model. The objective function value of the resulting LP relaxation is the lower bound on the total cost of the project.

We implement *Theorem 3* by using the LP based lower bound. To find UB, we use the heuristic procedure discussed in Section 3.3.4.2.

Iterative Application of the Elimination Rules

The mode elimination methods affect each other, because in eliminating short modes of an activity, all other activities are set to their longest modes and then the earliest start and latest completion times are computed. After some long modes are eliminated, the earliest start and latest completion times may change. Similarly after the short modes are eliminated, the earliest start and latest completion times by the long mode eliminations may change, since they rely on the shortest modes. Moreover LB_i value used in the costly mode eliminations is affected by any mode elimination, and affects the long mode eliminations once it eliminates short modes. The interactions between the elimination rules are summarized in Table 3-4.

As can be observed from the Table 3-4, applying these elimination methods iteratively, increases the total number of modes eliminated. The elimination procedure is implemented in the following manner. Initially long mode, short mode and costly mode iteration algorithms are applied in sequel. Then further iterations are performed according to their effects on the others. The table above indicates that the costly mode eliminations should be applied after any elimination, long mode elimination should be performed if any short mode is eliminated. There is no need to check short mode elimination unless a long mode is eliminated. The iterations stop when no further mode elimination can be done.

Table 3-4 – The Interactions between The Mode Elimination Rules

Affecting	Affected		
	Long Mode Elimination	Short Mode Elimination	Costly Mode Elimination
Long Mode Elimination	No	Yes	Yes
Short Mode Elimination	Yes	No	Yes
Costly Mode Elimination	Yes	No	Yes

3.2.3 The Optimal Linear Programming Relaxation Solution and Its Properties

We use the Linear Programming Relaxation (LPR), i.e., continuous relaxation, of the model to find lower bounds and upper bounds on the optimal total cost value.

Below we give an optimal LPR solution for a sample 4-activity project.

For $i = 1$; $j = \{1,2,3,4,5\}$ $X_{14} = 1$

For $i = 2$; $j = \{1,2,3,4,5\}$ $X_{21} = 0.76$ and $X_{25} = 0.24$

For $i = 3$; $j = \{1,2,3,4\}$ $X_{31} = 0.11$ and $X_{34} = 0.89$

For $i = 4$; $j = \{1,2\}$ $X_{42} = 1$

Note that the above optimal LPR solution gives at most two continuous variables for each activity. This property is inherent in all optimal LPR solutions as stated by the below property.

Property 1. The optimal solution of the LPR produces at most two continuous assignments for each activity.

Proof We let t_{iLP} denote the processing time assigned for activity i in the optimal LPR solution. Accordingly, $t_{iLP} = \sum_{j=1}^{m_i} t_{ij} X_{ij}$.

Given t_{iLP} values, for each activity i, the following LP model can be used to make minimum cost mode assignments.

$$Min \sum_{j=1}^{m_i} c_{ij} X_{ij}$$

$$s.to.$$

$$\sum_{\forall j} X_{ij} = 1$$

$$\sum t_{ij} X_{ij} = t_{iLP}$$

$$X_{ij} \geq 0$$

Since there are two constraints, there are two basic variables in all basic feasible solutions. This follows that each basic feasible solution has at most two nonnegative variables. From the LP theory, we know the optimal solution is in the basic feasible solution set, hence has at most two continuous variables. □

Hafızoğlu (2007) reports the same result for the general deadline problem.

Property 2. The continuous assignments associate to two consecutive modes for the activities having convex activity time/cost functions.

Proof Consider an activity *i*, having convex cost structure and two nonconsecutive modes a,b and $a < b$ as depicted by Figure 3-12.

Assume an optimal solution for the activity is on the line connecting modes *a* and *b*. As the time cost function is convex, any point on the lines connecting *a* and *a+1,* and *b-1* and *b* are below the line connecting *a* and *b*, i.e.,

$c_{ia}X_{ia} + c_{ib}X_{ib} > c_{ia}X_{ia} + c_{ia+1}X_{ib}$ and

$c_{ia}X_{ia} + c_{ib}X_{ib} > c_{ib-1}X_{ia} + c_{ia+1}X_{ib}$.

Hence any point on the line connecting two nonconsecutive modes cannot be optimal. In other words, an optimal solution is on the line connecting two consecutive modes, for all activities having convex time/cost structures. □

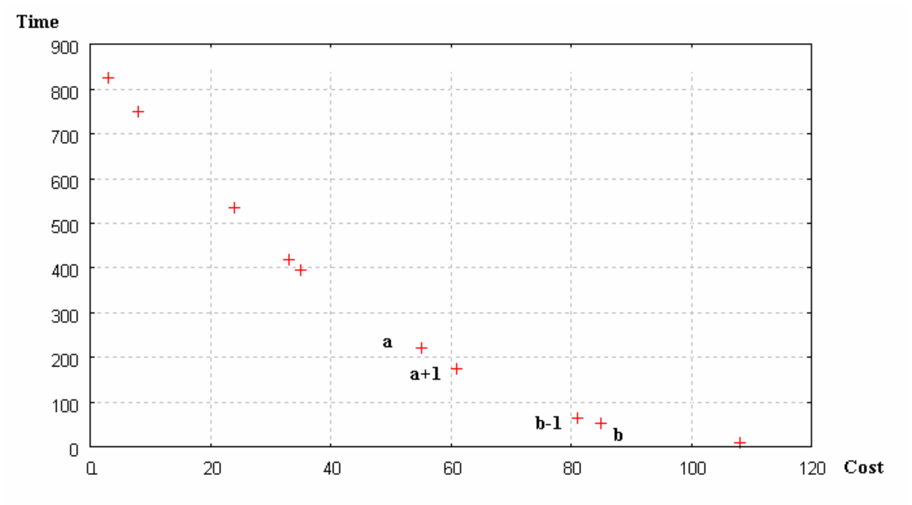

Figure 3-13 - A Sample Convex Time/Cost Function

According to the result of Property 2;

If $0 < X_{ij} < 1$ then,

Either $X_{ij+1} > 0$ and $X_{ij} + X_{ij+1} = 1$ or

$X_{ij-1} > 0$ and $X_{ij} + X_{ij-1} = 1$.

Property 3. The continuous assignments associate to two extreme modes, for the activities having concave time/cost functions.

Proof Consider an activity i having concave cost structure and two modes a,b such that $a < b$ and $a \neq 1$ and $b \neq m_i$ as depicted in Figure 3-14.

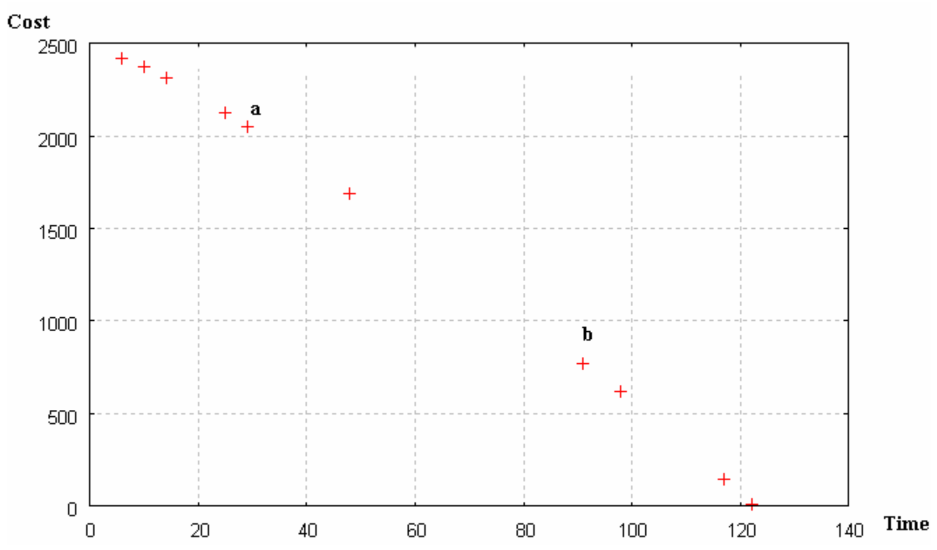

Figure 3-14 - A Sample Concave Time/Cost Function

Any point on the line connecting 1 and m_i is below the line connecting a and b., *i.e.*,

$c_{ia}X_{ia} + c_{ib}X_{ib} \geq c_{i1}X_{i1} + c_{im_i}X_{im_i}$

Hence any point on the line connecting two modes, other than the first and last modes, cannot be optimal. In other words, an optimal solution is on the line connecting the first and last modes, for all activities having concave time/cost structures. □

The result of Property 3 follows that if activity *i* has a concave time/cost function and $0 < X_{ij} < 1$ then $j = 1$ and $j = m_i$ and $X_{i1} + X_{im_i} = 1$.

Using the result of Property 3, we solve the LPR model only for the shortest and longest modes for the activities having concave time/cost functions.

Properties 4 through 7 state some properties of the optimal LPR solution for the general time/cost functions.

Property 4. If the optimal LPR solution produces fractional assignments for an activity, then that activity is critical for the LP relaxed problem.

Proof Assume an optimal LPR solution with $0 < X_{ij} < 1$, t_{iLP} is the associated duration for activity *i* and *B* is the associated total cost of the project.

If activity *i* is not critical, then there must be at least one $t_{ij'}$ such that $t_{ij'} > t_{iLP}$ and increasing the activity duration by $t_{ij'} - t_{iLP}$ units does not increase the total completion time of the project. Moreover, the total cost of the project, B', would be lower than B as $t_{ij'} > t_{iLP}$ implies $c_{ij'} < c_{iLP}$.

Since the optimal LPR model produces a total cost value higher than B', there cannot be such $t_{ij'}$ hence activity *i* is critical. □

Property 5. If an activity *i* is noncritical in the optimal LP solution then $X_{i1} = 1$.

Proof Any noncritical activity can be processed in at least one time period t such that $t > t_{iLP}$, without increasing the completion time of the project.

If such a t exists, then the solution is not optimal as the activity duration would increase by $t - t_{iLP}$ units. Such an increase would reduce the total cost which contradicts with the optimality of the solution. Hence there cannot be any $t > t_{iLP}$, and this follows $X_{i1} = 1$. □

Property 6. Increasing t_{iLP}, while keeping the processing times of all other activities *k* at t_{kLP} leads to an infeasible solution.

Proof Say processing activity *i* in time $t > t_{iLP}$ is considered.

Two cases exist.

Case 1: Activity *i* is critical. The processing time, t_{iLP}, cannot be increased without increasing the total completion time.

Case 2: Activity *i* is noncritical. Activity *i* is already executed at the longest duration mode. (See Property 5)

Hence, increasing the processing time of activity *i* makes the solution infeasible. □

Property 7. Decreasing t_{iLP} while keeping the processing times of other activities *k* at t_{kLP} leads to a feasible solution.

Proof Say processing activity *i* in time $t < t_{iLP}$ is considered.

Two cases exist.

Case 1: Activity *i* is critical. The processing time $t<t_{iLP}$ can be decreased, without increasing the total completion time of the project, hence the solution stays feasible.

Case 2: Activity *i* is noncritical. Activity *i* is already executed at the longest duration mode (see Property 5) and decreasing its processing time does not affect the total completion time of the project. □

3.2.4 Procedures for Finding Lower Bound

In this section, two different procedures are presented to find lower bounds on the optimal objective total cost values. The first lower bound that we refer to as naïve lower bound, is simple to compute, but inefficient. The second lower bound is based on the optimal LPR solutions. It is efficient, but harder to compute.

Naïve Lower Bound (NLB)

Naïve Lower Bound assigns all activities to their minimum-cost modes. The total cost of the resulting assignment is a lower bound on the total cost of the project. Formally;

$$NLB = \sum_{\forall i \in V} c_{i1}$$

NLB is a valid lower bound as any activity *i* cannot be executed with a cost that is smaller than c_{i1}, hence total project cost cannot be lower than $\sum_{\forall i \in V} c_{i1}$. The resulting schedule on the hand, may be infeasible, as it leads to the maximum duration assignments, hence high project completion times (higher than T_{min}).

A very simple example below illustrates NLB. The project consists of 4 activities. Modes of each activity are given below.

For $i = 1$, there are 5 modes, with the following time/cost pair set:

$\{(108,10),(85,537),(81,625),(61,1050),(55,1164)\}$

For $i = 2$, there are 5 modes, with the following time/cost pair set:

$\{(99,15),(93,140),(80,407),(66,666),(61,750)\}$

For $i = 3$, there are 4 modes, with the following time/cost pair set:

$\{(104,11),(71,313),(68,331),(49,415)\}$

For $i = 4$, there are 2 modes, with the following time/cost pair set:

$\{(88,15),(35,99)\}$

The immediate precedence relations are tabulated below.

Table 3-5 - The Precedence Relations for the Example

Activity	Immediate Predecessors
1	4
2	-
3	4
4	-

NLB is found by simply taking the minimum cost mode for each activity. Accordingly, $NLB = \sum_{\forall i \in V} c_{i1} = 10 + 15 + 11 + 15 = 51$

In our experiments, we use the naïve lower bound as filtering mechanisms. We first compute naïve lower bound and compute LPR Based Lower Bounds if the naïve lower bound cannot eliminate.

LPR Based Lower Bound (LPLB)

LPLB is found by relaxing the integrality constraints on the discrete variables, X_{ij} s, and solving the resulting LP to optimality. An optimal solution to any relaxation, hence the LPR, provides a lower bound on our minimization problem.

NLB is a lower bound on the optimal solution of the LPR as well. Hence LPLB dominates NLB, but at an expense of additional effort.

$LPLB = 1823.01$ (found by the LP software), for the sample project whose NLB was 51.

3.2.5 Solution Algorithms

We first reduce the size of the minimum deadline problem by an iterative application of the mode elimination rules, and then solve the reduced problem by branch and bound algorithm and approximation procedures. In this section we discuss our solution procedures.

3.2.5.1 Branch and Bound Algorithm

Our branch and bound algorithm uses two different branching strategies. Two types of routing strategies are used in each branching strategy. Both strategies are based on the optimal LPR solution.

Branching Strategy 1

Branching strategy is based on the LP relaxation (LPR) solution. Stepwise description of the strategy is presented below.

Step 0. Solve the LPR problem at the root node.
Use LP based heuristic algorithm (discussed in Section 3.2.5.2) to find an initial feasible solution. Let the UB be the total cost value of the solution.
Recall from Property *1* that the optimum LPR solution has at most two fractional variables for each activity.

Step 1. The variable with the highest fractional value, say X_{ij}, is chosen.

Step 2. There are two branches associated with the selected variable. These branches are $X_{ij} = 1$ and $X_{ij} = 0$ as shown in the following figure.

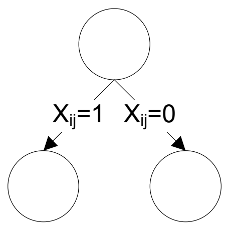

Figure 3-15 - Branching Strategy 1

Two different routing strategies are proposed.

Routing Strategy 1: The branch that set the most fractional variable to 1 is followed.

Routing Strategy 2: Lower bound is used to find the route. LPR is solved for both branches. If both routes are feasible then the one which gives the smaller lower bound is followed. If both are infeasible then go to *Step 4*.

Step 3. Solve the LPR problem.

There are three cases.

Case i. LP is infeasible, then go to *Step 4.*

Case ii. All variables in the LPR solution are integer, the solution is optimal for the associated node. If $UB > LB$ (LB is the optimal LPR cost) then let $UB = LB$. Go to Step 4.

Case iii. There are two sub-cases.

a. LPR is feasible and fractional and $UB \geq LB$ then go to Step 1.
b. Fathoming case: $UB < LB$, then go to Step 4.

Step 4. Backtrack to the parent node of the branch and bound tree. Select the alternate branch, go to *step 0*. If the alternate branch is previously visited then backtrack to the grand parent node, so on, until finding an unvisited branch or reaching the root node. If the current node is the root node and both child nodes are visited then the solution giving UB, i.e., the best upper bound is the optimal solution.

Branching Strategy 2

This strategy is also based on the optimal LPR solution; however rather than two branches, we consider three branches at each node. The stepwise description of the procedure is provided below.

Step 0. Solve the LPR problem at the root node. Use LP based heuristic algorithm (discussed in section 3.2.5.2) to find an initial feasible solution. Let upper bound be the total cost value of the solution.

Step 1. Find the activity that has the most fractional valued variable. Considering $\sum_{\forall j} X_{ij} = 1$ and Property 1, the most fractional and least fractional variables in the problem correspond to the same activity. Let X_{ij}^{Min} and X_{ij}^{Max} be least and most fractional variables respectively.

Step 2. There are three branches.

First branch: $X_{ij}^{Min} = 1$ and $X_{ij}^{Max} = 0$.

Second branch: $X_{ij}^{Min} = 0$ and $X_{ij}^{Max} = 1$.

Third branch: Both variables are set to be zero: $X_{ij}^{Min} = 0$ and $X_{ij}^{Max} = 0$.

The following figure illustrates the branching scheme.

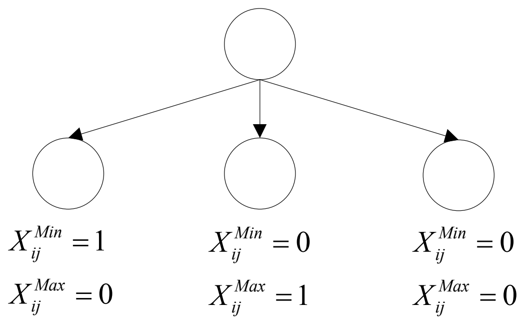

Figure 3-16 - Branching Strategy 2

If an activity has only two modes, then there are two branches.

The routing strategies that are defined for *branching strategy 1* can be used to find the route.

Step 3. Solve the LPR problem. There are three alternative cases. If none of the conditions stated in the cases are satisfied then go to *Step 1.*

Case i. LP is infeasible, then go to *Step 4.*

Case ii. All variables in the optimal LPR solution are integer; the solution is optimal for the associated node. If $UB > LB$ (LB is the optimal LPR cost) then let $UB = LB$. Go to Step 4.

Case iii. There are two sub-cases.

a. LPR is feasible and fractional and $UB \geq LB$ then go to Step 1.

b. Fathoming case: $UB < LB$, then go to Step 4.

Step 4. Backtrack to the parent node of the branch and bound tree. Select the alternate branch, go to *step 0*. If the alternate branch is previously visited then backtrack to the grand parent node, and so on until finding an unvisited route or reaching the initial node. If the current node is the root node and all three child nodes are visited then the solution giving UB, i.e., UB the best upper bound is the optimal solution.

An example to illustrate the branching strategies is provided Table 3-6. Time/cost values of the modes are stated in the parentheses. The first number in each parenthesis is the processing time, while the second one is the cost of the activity.

The fractional variables of the optimal LPR solution of the root node are represented in Table 3-7 below.

Branching s*trategy 1:* The index of the variable with the highest fractional value is 14-0. If *routing strategy 1* is utilized to find the route then $X_{14,0} = 1$. If *routing strategy 2* is applied then $X_{14,0} = 0$ because when $X_{14,0} = 1$ the problem is infeasible. If $X_{14,0} = 1$ led to a feasible solution then the total costs by the optimal LPR solutions would be compared and the route having the smaller lower bound would be selected.

Branching strategy 2: Two branches $X_{14,0} = 1$; $X_{14,1} = 0$ and $X_{14,0} = 0$; $X_{14,1} = 1$ lead to infeasible solutions. The remaining branch $X_{14,0} = 0$, $X_{14,1} = 0$ is feasible solution and hence it is selected. If the other branches were also feasible the total cost by the optimal LPR solutions would be compared and the route having the smaller lower bound would be selected.

Table 3-6 – A Sample Project to Illustrate Branching Strategies

Activity	Immediate Successors	Modes
0	1,2,3	(0,0)
1	4,5	(35,1536)(22,1570)(24,1698)(8,1896)(3,1953)
2	11,12	(99,15)(93,140)(80,407)(66,666)(61,750)(38,1072)
3	23	(104,11)(71,313)(68,331)(49,415)
4	6,7	(88,15)(35,99)
5	11,12	(82,9)(57,236)(35,373)(4,546)
6	8,9	(108,5)(84,527)(77,678)(40,1386)(30,1565)(24,1672)
7	11,12	(112,13)(46,156)
8	10	(58,1622)(40,2005)(38,2042)(21,2340)
9	11,12	(117,8)(95,524)(78,906)(72,1023)(67,1115)(33,1673)(21,1847)(10,1998)
10	11,12	(83,8)(79,74)(65,263)(52,413)(44,503)(23,707)
11	13,14	(111,11)(52,682)(13,1071)(6,1133)
12	18,19	(117,15)(112,95)(109,134)(59,709)(41,898)(26,1052)(11,1180)
13	15,16	(85,7)(76,83)(73,99)(20,265)
14	18,19	(100,14)(38,124)
15	17	(91,5)(40,193)
16	18,19	(118,10)(111,180)(94,586)(86,756)(80,867)(64,1150)(62,1182)(42,1488)(32,1635)
17	18,19	(121,5)(117,88)(110,220)(99,398)(87,580)(66,842)(62,888)(38,1131)
18	20,21	(112,12)(91,581)(84,766)(80,862)(76,948)(70,1066)(62,1120)(49,1439)(30,1719)
19	23	(53,8)(48,45)(12,216)(6,228)
20	22,23	(91,15)(84,210)(82,263)(78,363)(72,496)(62,694)(25,1392)(16,1537)(8,1648)
21	23	(110,7)(97,37)
22	23	(63,15)(43,53)
23	-	-

Table 3-7 – The Variables at Fractional Values in the Optimal LPR Solution

Variable Index (i-j)	Value
7-0	0.33
7-1	0.67
9-0	0.32
9-7	0.68
12-0	0.88
12-6	0.12
14-0	0.97
14-1	0.03
16-0	0.53
16-8	0.47
20-0	0.55
20-8	0.44

3.2.5.2 Approximation Algorithm

We propose three heuristic procedures to find feasible approximate solutions to our problem. The first heuristic is LP based, runs in polynomial time and is used as an initial feasible solution in our branch and bound algorithm. The second heuristic is branch and bound based, runs in polynomial time and is proposed with the hope of obtaining high quality and fast solutions. The third heuristic is also branch and bound based, runs in exponential time and proposed with the hope of obtaining near optimal solutions.

LP Based Heuristic Algorithm

Our heuristic procedure to find an upper bound proceeds in two phases: construction and improvement.

Construction Phase:

The construction phase consists of 2 steps.

Step 1. The problem is first solved by relaxing the integrality constraints. Say t_{iLP} denotes the processing time of activity and c_{iLP} is its associated cost in the optimal relaxed solution.

Step 2. The total completion time is kept at its minimum value of T_{Min}, when the processing time of activity i is decreased from t_{iLP}. We decrease the processing time of activity i by setting it to the largest processing time mode that is no larger than t_{iLP}. (From Property 6 we know that decreasing any processing time does not violate feasibility.) Our aim is to increase the total cost as small as possible by small movements.

Consider the sample project whose data were given in Table 3-6. In the construction phase, the following mode assignments are made and the associated upper bound is found as 16549.

For activity 7, $t_{7LP} = 68$ and the mode with minimum duration that is not larger than 68 is mode 2, hence we set $X_{7,2} = 1$.

For activity 9, $t_{9LP} = 44$, and the mode with minimum duration that is not larger than 44 is mode 6, hence we set $X_{9,6} = 1$.

For activity 12, $t_{12LP} = 104$, and the mode with minimum duration that is not larger than 104 is mode 4, hence we set $X_{12,4} = 1$.

For activity 14, $t_{14LP} = 98$, and the mode with minimum duration that is not larger than 98 is mode 2, hence we set $X_{14,2} = 1$.

For activity 16, $t_{16LP} = 78$, and the mode with minimum duration that is not larger than 78 is mode 5, hence we set $X_{16,5} = 1$.

For activity 20, $t_{20LP} = 54$, and the mode with minimum duration that is not larger than 54 is mode 6, hence we set $X_{20,6} = 1$.

The other variables at the optimal solution are not fractional. Hence the solution obtained after the above shifts is feasible for the original problem, and the objective function value is the upper bound for the total cost of the problem.

Improvement Phase:

The solution found by the construction phase is improved using two procedures in sequel.

First Improvement

The construction heuristic can be improved by increasing the durations of some activities. Such increases are possible, as in construction phase we decrease some durations that would give room to some other increases. For each noncritical activity, we check whether it can be moved to its next higher duration mode without violating feasibility. Among the activities that would lead to feasible increases, we select the one that causes the maximum reduction in the total cost value. Formally, we select the activity k such that $c_{kj} - c_{kj-1} = \max_{i} \left\{ c_{ij} - c_{ij-1} \right\}$ if activity i is assigned to mode j in the current solution. We terminate the first improvement whenever any further increase in durations leads to an infeasible solution.

For the sample project used to illustrate the construction phase, activity 22 is assigned to mode 2, which has a cost of 53. Assigning activity 22 to mode 1, which is the next higher duration mode, produces a feasible solution hence the activity 22 is moved to its first mode. This movement decreases the total cost of the project by 38 units, and the new total cost is 16511.

We terminate the first improvement step, as any further increase in durations leads to an infeasible solution.

Second Improvement

The second improvement phase tries to improve the solution returned by the first improvement phase using pairwise interchanges. For all pairs on the same path, we check whether increasing one activity duration while decreasing the other one reduces the total cost without increasing the project completion time. Among the pairs that qualify, we select the interchange that decreases the total cost by the largest amount. We terminate whenever all pairwise interchanges are either infeasible or cannot lead to a reduction in total cost.

Branch and Bound Based Approximation Algorithm I

To find second approximate solution, we use our branch and bound algorithm with the first branching strategy, but do not allow backtracking. We use the LPR based heuristic as a starting solution in our curtailed branch and bound algorithm. We terminate whenever we find all integer variables, no feasible solution, or no promising solutions (lower bounds are no smaller than the best known upper bound). For the sake of completeness *Step 0* and *Step 1* are restated below, although they are the same as in optimization algorithm. In *step 2*, *routing strategy 2* defined in the previous section is used.

Step 0. Solve the LPR problem at the root node. Use LP based heuristic algorithm to find an initial feasible solution. Let upper bound be the total cost value of the solution. Recall from Property 1 that the optimum LPR solution has at most two fractional variables for each activity.

Step 1. The variable with the highest fractional value is chosen. Say X_{ij} is the most fractional variable.

Step 2. There are two branches associated with the selected variable. These branches are $X_{ij} = 1$ and $X_{ij} = 0$ as shown in the following figure.

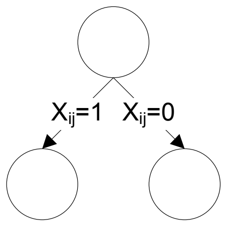

Figure 3-17 - Branching Strategy 1

Lower bound is used to find the route. LPR is solved for both branches. If both routes are feasible then the one with the smaller lower bound is followed. If both are infeasible then go to *Step 3*.

Step 3. Solve the LPR problem. There are three cases.

Case i. LP is infeasible, then stop. The solution giving the best upper bound is the solution of approximation algorithm.

Case ii. All variables in the optimal LPR solution are integer; the solution is optimal for the associated node. If $UB \geq LB$ (LB is the total cost of found in optimal LPR solution) then let $UB = LB$. Then stop. The solution giving the best upper bound is the solution of approximation algorithm.

Case iii. There are two sub-cases.

a. LPR is feasible and fractional and $UB \geq LB$, then go to Step 1.
b. $UB < LB$ *,* then stop. The solution giving the best upper bound is the result of approximation algorithm.

In place of *Branching Strategy 1, Branching Strategy 2* could also be used to arrive at an approximate solution.

Branch and Bound Based Approximation Algorithm II

In this section we describe another approximation algorithm that is based on reducing the problem size using the optimal LPR solution and then solving the reduced problem by our branch and bound algorithm. In doing so, the variables that take integer values in the optimal LPR solution are fixed and then branch and bound is applied for the fractional variables. The method consists of the following three steps.

Step 0. The LPR problem is solved.

Step 1. All variables that take integer values in the optimal LPR solution are fixed.

Step 2. Branch and bound algorithm is used to find the values of the activities with fractional assignments.

3.3 General Modified Budget Problem

In this section, we consider a general budget problem, with available budget b, between b_{min} and b_{max} values (See Section 3.1.2.). We aim to select the solution having the smallest total cost value (no more than b) among the ones having the smallest project completion time value.

Recall that, when $b = b_{min}$ the problem is trivial. All activities are assigned to their longest modes. For $b = b_{max}$ the minimum deadline problem is already defined and studied as a special case of the budget problem. We now discuss general budget problem with arbitrary b value, where $b_{min} < b < b_{max}$. In the next subsection mode elimination procedures are proposed. In Section 3.3.2 we state the properties of the optimal LP relaxation (LPR). We continue with lower bounding procedures. The branch and bound method and approximation algorithms are presented in Section 3.3.4. The majority of the results are generalized from the minimum deadline problem. For the sake of simplicity, hereafter we refer to 'the general modified budget problem' as 'the budget problem'

3.3.1 Mode Elimination Rules

In this section we derive short mode, long mode and costly mode elimination rules for the budget problem. Our aim is to reduce the size of the search by eliminating some modes that are either infeasible or cannot lead to an optimal solution.

Short Mode Elimination

As the short modes are costly, assigning these modes may lead to infeasible solutions. We use the following notation in the algorithm.

ES_L_i: Earliest start time of activity *i*, when all activities are assigned to their longest modes.

$LC\_L(LB)_i$: Latest completion time of activity *i*, when all activities are assigned to their longest modes. A lower bound for the project completion time is used to find the $LC\_L(LB)_i$, as the exact completion time is not known.

ES_L_i and $LC\_L(LB)_i$ are found by the CPM method, using a lower bound on the completion time of the project.

While computing ES_L_i and $LC\_L(LB)_i$ we assign all noncritical activities to their longest modes, i.e., we set

$X_{i1} = 1$ for all *i*.

Theorem 4

If $LC\_L(LB)_i - ES\_L_i \geq t_{ij}$ then for activity *i*, modes $j+1$ through m_i cannot lead to an optimal solution.

Proof: Assume a solution that contradicts with the condition of the theorem, i.e., $LC\_L(LB)_i - ES\_L_i \geq t_{ij}$ and $X_{ij'} = 1$ where $j' > j$. Since $c_{ij'} > c_{ij}$ setting $X_{ij} = 1$ is certainly feasible and improves the budget by an amount of $c_{ij'} - c_{ij}$. And such an interchange is feasible as $LC\_L(LB)_i - ES\_L_i \geq t_{ij}$ Hence any solution that contradicts with the condition of the theorem cannot be optimal. □

The example that associated information is given in Table 3-6 is used to illustrate the application of the rule. Let the available budget for the project be 8371 YTL. The lower bound for the total project completion time is found to be 517.64 by the CPM method. Consider activity 3 that has neither a successor nor a predecessor. Hence;

$LC\_L(LB)_3 = 517$

$ES\_L_i = 0$

Since $517 \geq t_{3,1} = 104$, all modes $j \in M_i$ that are $j > 1$, i.e., mode 2, mode 3 and mode 4 are eliminated. Since mode 1 is feasible in any solution and other modes are more costly than mode 1.

Theorem 4 cannot eliminate any mode of any activity other than the modes stated above.

Long Mode Elimination

The long mode elimination rule eliminates all nonpromising modes, i.e., the ones that cannot produce optimal solutions. Recall that in the minimum deadline problem, long mode elimination rule disregards the infeasible solution as the project completion time is a constraint, but do not appear in the objective function. We use the following notation to state long mode elimination algorithm.

ES_E_i: Earliest start time of activity *i*, when all activities are assigned to their shortest modes.

$LC\_E(UB)_i$: Latest completion time of activity *i*, when all activities are assigned to their shortest modes. Upper bound for total completion time of the project is used to find $LC\_E(UB)_i$ as the project completion time is not known. The procedure for finding the upper bound will be discussed later.

In computing the ES_E_i and $LC\_E(UB)_i$ values, the CPM method is used by setting all activities to their shortest modes, i.e. we set

$X_{im_i} = 1$ for all *i.*

Theorem 5

If $LC\_E(UB)_i - ES\_E_i < t_{ij}$ for activity *i* then mode 1 through j cannot lead to an optimal solution.

Proof: Consider an optimal solution where S^*_{N+1} is the associated project completion time, S^*_i is the start time of the activity *i*, LC^*_i is the latest completion time of the activity.

$S^*_i \geq ES\_E_i$ since $t^*_k \geq t_{km_k}$ where *k* is the immediate predecessor of activity *i* and t^*_k is its optimal processing time.

$LC^*_i \leq LC\_E(UB)_i$ because $S^*_{N+1} < UB$ and $t^*_k \geq t_{km_k}$.

This follows $LC^*_i - S^*_i \leq LC\_E(UB)_i - ES\_E_i$. Any solution that assigns activity *i* to mode *j* such that $t_{ij} > LC\_E(UB)_i - ES\_E_i$ cannot be optimal as the condition $LC^*_i - S^*_i \geq t_{ij}$ is violated. □

We illustrate the application of the theorem with the example that associated information is given in Table 3-5. Budget limit for the problem is set to 1070 YTL. The upper bound for the project completion time is 116. $LC\_E(UB)_1$ and ES_E_1 are computed using the CPM method.

$LC\_E(UB)_1 = 81$

$ES\_E_1 = 0$

$t_{1,1} = 108 > 81$ and $t_{1,2} = 85 > 81$ therefore mode 1 and mode 2 defined for activity 1 cannot be in any optimal solution and are eliminated. Similarly first modes of activity 3 activity 4 are eliminated.

Costly Mode Eliminations:

The costly mode elimination rule eliminates the modes that would lead to an infeasible solution, without ever being evaluated by a solution algorithm. Recall that the costly mode elimination rule eliminates nonpromising modes in the minimum deadline problem as the budget is not a constraint, but appears in the objective.

LB_i: A valid lower bound on the total cost of all activities, other than *i*.

Theorem 6

If $LB_i + c_{ij} > b$ then modes 1 through j cannot lead to a feasible solution.

Proof: LB_i is the minimum cost of performing all other activities, except activity *i*. When activity *i* is assigned to any mode $j \in M_i$ then the lower bound on total cost becomes $LB_i + c_{ij}$ and if $LB_i + c_{ij} > b$, assignment to mode *j* violates the budget constraint. □

LB_i used in Theorem 6 is computed in two different ways.

Simple Lower Bound

A lower bound on the total cost for the activities except activity *i* is the cost incurred by these activities when they are assigned to their longest modes.

$$NLB_i = \sum_{k \in V, k \neq i} c_{k1}$$

LP Based Approach for Finding Lower Bound

The procedure consists of two steps.

Step 1:

Activity *i* is fixed to its shortest duration mode and its cost is set to zero, i.e., $X_{im_i} = 1$ and $c_{im_i} = 0$.

Step 2:

The discrete nature of X_{ij} variables is relaxed and the problem is solved as a continuous LP model. The objective function value of the resulting LP relaxation is the lower bound on the total cost of the project.

Iterative Application of the Elimination Rules

As discussed for the minimum deadline problem, the mode elimination methods affect each other. The interactions are the same as those that are defined for minimum deadline problem and are restated in Table 3-8 for the sake of completeness.

Table 3-8 – The Interactions between Mode Elimination Rules

Affecting	Affected		
	Long Mode Eliminations	Short Mode Eliminations	Costly Mode Eliminations
Long Mode	No	Yes	Yes
Short Mode	Yes	No	Yes
Costly Mode	Yes	No	Yes

Due to the effects of elimination algorithms to each other, iterative application of these algorithms increases the effectiveness of the algorithms. Initially long mode, short mode and costly mode iteration algorithms are applied in sequel. Then further iterations are performed according to the effects of methods to each other. The table above indicates that the costly mode elimination algorithm should be applied after any elimination, long mode elimination should be performed if a short mode or costly mode is eliminated. There is no need to check short mode elimination unless a long mode is eliminated. The iterations should stop if no further mode elimination is possible.

3.3.2 The Optimal Linear Programming Relaxation Solution and Its Properties

We use Linear Relaxation programming (LPR), i.e., continuous relaxation, of the model to find lower bounds and upper bounds on the optimal objective function value. The deadline problem and the budget problem are analogous in the sense that the objective function in one problem appears as a constraint in the other and vice versa. Hence most of the properties we derived for the minimum deadline problem hold for the budget problem. We restate the properties and modify their proofs so that they treat the budget as a constraint, but not an objective function.

Property 8. The solution of the optimal LPR produces at most two continuous assignments for each activity.

Proof: We let c_{iLP} and t_{iLP} denote the cost and duration assigned for activity *i* respectively. Accordingly,

$c_{iLP} = \sum_{\forall j \in M_i} c_{ij} X_{ij}$ and $t_{iLP} = \sum_{\forall j \in M_i} t_{ij} X_{ij}$

Given c_{iLP} and t_{iLP} values for each activity i, the following LP model can be used to make mode assignments with the minimum cost.

$$Min \sum_{j=1}^{m_i} t_{ij} X_{ij}$$

$$s.to.$$

$$\sum_{\forall j} X_{ij} = 1$$

$$\sum_{\forall j \in M_i} c_{ij} X_{ij} = c_{iLP}$$

$$X_{ij} \geq 0$$

Since there are two constraints, there are two basic feasible variables in the basic feasible solution. This follows each basic feasible solution has at most two nonnegative variables. From LP theory, we know the optimal solution is the basic feasible solutions' set. Therefore the optimal LP solution has at most two positive X_{ij} values. □

Property 9. The continuous assignments associate to two consecutive modes for the activities having convex time/cost functions.

Proof: Consider an activity i having convex cost structure and two nonconsecutive modes $a,b \in M_i$ and $a < b$, as depicted by the Figure 3-.

Assume an optimal solution for the activity is on the line connecting *a* and *b*. As the time/cost function is convex, any point on the lines connecting *a* and *a+1,* and *b-1* and *b* are below the line connecting *a* and *b*.

$t_{ia}X_{ia} + t_{ib}X_{ib} > t_{ia}X_{ia} + t_{ia+1}X_{ib}$ and

$t_{ia}X_{ia} + t_{ib}X_{ib} > t_{ib-1}X_{ia} + t_{ia+1}X_{ib}$.

Hence any point on the line connecting two nonconsecutive modes cannot be optimal. In other words, the optimal solution is on the line connecting two consecutive modes, for all activities having convex time/cost structures. □

The result of Property 9 follows that, if $0 < X_{ij} < 1$ then,

Either $X_{ij+1} > 0$ and $X_{ij} + X_{ij+1} = 1$ or

$X_{ij-1} > 0$ and $X_{ij} + X_{ij-1} = 1$.

Property 10. The continuous assignments associate to two extreme modes, the activities having concave time/cost functions.

Proof: Consider an activity *i* having concave cost structure and two modes $a,b \in M_i$, $a < b$ and $a \neq 1$ and $b \neq m_i$ as depicted in Figure 3-. Any point on the line connecting 1 and m_i is below the line connecting *a* and *b, i.e.,*

$t_{ia}X_{ia} + t_{ib}X_{ib} \geq t_{i\underline{m}_i}X_{i\underline{m}_i} + t_{i\overline{m}_i}X_{i\overline{m}_i}$

Hence any point on the line connecting two modes, other than the first and last modes, cannot be optimal. In other words, an optimal solution is on the line connecting first and last modes for all activities having concave time/cost structures. □

The result of Property 10 follows that if $0 < X_{ij} < 1$ then $j = 1$ and $j = m_i$ and $X_{i1} + X_{im_i} = 1$.

Using the result of Property 10, we solve the LPR model only for the shortest and longest modes for the activities having concave time/cost functions.

Properties 11 through 14 state some properties of the optimal LPR solution for the general time/cost functions.

Property 11. If the optimal LPR solution produces fractional assignments for an activity, then that activity is critical for the LP relaxed problem.

Proof: Assume an optimal LPR solution with $0 < X_{ij} < 1$, t_{iLP} is the associated duration for activity *i* and *B* is the associated total cost of the project.

If activity *i* is not critical, then there must be at least one $t_{ij'}$ such that $t_{ij'} > t_{iLP}$ and increasing the activity duration by $t_{ij'} - t_{iLP}$ units does not increase the total completion time of the project. Moreover, the total cost of the project, B', would be lower than B as $t_{ij'} > t_{iLP}$ implies $c_{ij'} < c_{iLP}$.

Since the LPR model produces a total cost value higher than B', there cannot be such $t_{ij'}$ hence activity *i* is critical. □

Property 12. If an activity *i* is noncritical in the optimal LP solution then $X_{i1} = 1$.

Proof: Any noncritical activity can be processed in at least one time period t such that $t > t_{iLP}$, without increasing the completion time of the project.

If such a t exists, then the solution is not optimal as the activity duration would increase by $t - t_{iLP}$ units. Such an increase would reduce the total cost which contradicts with the optimality of the solution. Hence there cannot be any $t > t_{iLP}$, and this follows that $X_{im_i} = 1$. □

Property 13. Decreasing t_{iLP} while keeping processing times of all other activities *k* at t_{kLP} always leads to an infeasible solution.

Proof: $b < b_{max}$ and $S^*_{N+1} > T_{Min}$ where T_{Min} is the minimum possible project duration and S^*_{N+1} is the project duration at optimal. If $B^* < b$, where B^* is the total project cost at optimal then S^*_{N+1} can be improved at a cost of $b - B^*$. Since S^*_{N+1} is the minimum project duration subject to available budget, $B^* = b$. Decreasing t_{iLP} leads to an increase in total project cost, B^*. If B^* increases further then the problem becomes infeasible. □

Property 14. Increasing t_{iLP} while keeping the durations of all other activities *k* at t_{kLP} leads to a feasible solution.

Proof: Say activity *k* is assigned to mode j, where $t_{kj} > t_{kLP}$ is considered.

As $c_{kj} < c_{kLP}$,

$$\sum_{\forall i \in V, i \neq k} c_{iLP} + c_{kj} < \sum_{\forall i \in V, i \neq k} c_{iLP} + c_{kLP} = b\,. \qquad \square$$

3.3.3 Procedures for Finding Lower Bound

In this section two different procedures are presented to find lower bounds on the optimal objective function values, i.e., the optimal project completion time, in this section. The first lower bound that we refer as naïve lower bound is simple to compute but inefficient. The second lower bound is based on the optimal LPR solutions. It is efficient but harder to compute.

Naïve Lower Bound (NLB)

NLB is found by setting all activities to their shortest duration modes. The resulting project completion time, NLB found by the CPM method, is a lower bound for the optimal project completion time.

Formally;

$NLB = S_{N+1}$ where $X_{im_i} = 1$ for $\forall i \in V$

NLB is a valid lower bound because any activity *i* cannot be executed with in a duration that is smaller than t_{im_i} and hence total project completion time cannot be less than *NLB*. The resulting schedule on the other hand, may be infeasible, as it leads minimum duration assignments, leading to high project cost (higher than *b*).

An example whose data are given in Table 3-5 is used to illustrate the NLB computation.

NLB is found by simply taking the minimum duration mode for each activity. Accordingly, as activity *3* and activity *4* are on the critical path, the total completion time is the sum of processing times of these activities. Accordingly, $NLB = S_{N+1} = 49 + 35 = 84$.

LPR Based Lower Bound (LPLB)

LPLB is found by relaxing the integrality constraints on the discrete variables, X_{ij} s, and solving the resulting LP to optimality. An optimal solution to the LPR of the problem provides a lower bound for our minimization problem.

NLB is a lower bound on the optimal solution of the LPR as well. Hence LPLB dominates NLB.

The LPLB value is 116 (found by the LP software) for the sample project above whose NLB is 84.

3.3.4 Solution Algorithms

We first reduce the size of the budget problem by an iterative application of the mode elimination rules, then solve the reduced problem by branch and bound algorithm and approximation procedures. In this section we describe our optimization algorithms and approximation algorithms for the general modified budget problem.

3.3.4.1 Branch and Bound Algorithm

The branch and bound algorithm described for the minimum deadline problem is used for the general budget problem. As in the deadline problem two different branching strategies together with two types of routing strategies are presented.

Branching Strategy 1

Branching strategy is based on the LP relaxation (LPR) solution. Stepwise procedure for the strategy is presented below.

Step 0. Solve the LPR problem at the root node.

Use LP based heuristic algorithm (discussed in Section 3.3.3) to find an initial feasible solution. Let upper bound be the total completion time of the solution and B^* be its total cost value.

Recall from Property 8 that LPR solution has at most two fractional variables for each activity.

Step 1. The variable with the highest fractional value, say X_{ij}, is chosen.

Step 2. There are two branches associated with the selected variable. These branches are $X_{ij} = 1$ and $X_{ij} = 0$ as shown in the following figure.

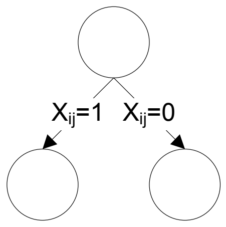

Figure 3-18 - Branching Strategy 1

Considering these two branches at each node, two different routing strategies are proposed.

Routing Strategy 1: Most fractional variable is set to be one.

Routing Strategy 2: Lower bound is used to find the route. LPR is solved for both branches. If both routes are feasible then the one which gives the smaller lower bound is followed. If both are infeasible then go to *Step 4*.

Step 3. Solve LPR. There are three alternative cases.

Case i. LP is infeasible, then go to *Step 4.*

Case ii. All variables in the LPR solution are integer, the solution is optimal for the associated node. Let B be the total cost value of the LPR solution. There are two cases.

a. $UB > LB$ (*LB* is the optimal LPR cost) then let $UB = LB$, $B^* = B$.
b. $UB = LB$. If $B < B^*$ then $B = B^*$. Go to Step 4.

Case iii. There two cases.

a. LPR is feasible and fractional and $UB \geq LB$, then go to Step 1.
b. Fathoming case: $UB < LB$, then go to Step 4.

Step 4. Backtrack to the parent node of the branch and bound tree. Try the alternate branch, go to *step 0*. If the alternate branch is previously visited then backtrack to the grand parent node, so on, until finding an unvisited route or reaching the root node. If the current node is the root node and both child nodes are visited then the solution giving UB, i.e., the best upper bound is the optimal solution.

Branching Strategy 2

This strategy is also based on the optimal LPR solution; however in place of two there are three branches at each node. Stepwise procedure is provided below.

Step 0. Solve the LPR problem at the root node. Use LP based heuristic algorithm to find an initial feasible solution. Let upper bound be the total completion time of the solution and B^* be its total cost value.

Step 1. Find the activity that has the variable with the most fractional value. Considering $\sum_{\forall j} X_{ij} = 1$ and Property 8 of the optimal LPR solution, the most fractional and least fractional variables in the problem correspond to the same activity. Let X_{ij}^{Min} and X_{ij}^{Max} be least and most fractional variables respectively.

Step 2. There are three branches.

First Branch: $X_{ij}^{Min} = 1$ and $X_{ij}^{Max} = 0$.

Second Branch: $X_{ij}^{Min} = 0$ and $X_{ij}^{Max} = 1$.

Third Branch: Both variables are set to be zero: $X_{ij}^{Min} = 0$ and $X_{ij}^{Max} = 0$.

The following figure illustrates the branching scheme.

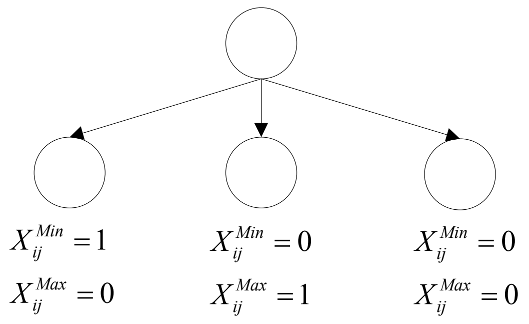

Figure 3-19 - Branching Strategy 2

If an activity has only two modes, then there are two branches.

The same routing strategies that are defined for *branching strategy 1* can be used to find the route.

Step 3. Solve the LPR problem. There are three alternative cases. If none of the conditions stated in the cases are satisfied then go to *Step 1.*

Case i. LP is infeasible, then go to *Step 4.*

Case ii. All variables in the optimum LPR solution are integer, the solution is optimal for the associated node. If $UB > LB$ (LB is the total completion time of the optimal LPR solution) then let $UB = LB$. Go to Step 4.

Case iii. There are two sub-cases.

a. LPR is feasible and fractional and $UB \geq LB$, then go to Step 1.
b. Fathoming Case: $UB < LB$ *,* then go to Step 4.

Step 4. Backtrack to the parent node of the branch and bound tree. Try the alternate branch, go to *step 0*. If the alternate branch is previously visited then backtrack to the grand parent node, and so on until finding an unvisited route or reaching the initial node. If the current node is the root node and all three child nodes are visited then the solution giving UB, i.e., UB the best upper bound is the optimal solution.

An illustrative example for both strategies is provided below. The associated information is given in Table 3-6. Applying the procedures described above in the LPR solution at the root node, there are the fractional variables shown in Table 3-9.

Table 3-9 – The Variables That Take Fractional Values in LPR Solution

Variable Index (i-j)	Value
17-0	0.48
17-7	0.51
18-0	0.36
18-8	0.63
20-0	0.71
20-8	0.29

Branching s*trategy 1*: The index of the variable with the highest fractional value is 20-0. If *routing strategy 1* is applied to find the route then $X_{20,0} = 1$. If *routing strategy 2* is applied then again $X_{20,0} = 1$; because when both $X_{14,0} = 0$ and $X_{14,0} = 1$ are feasible, but when $X_{14,0} = 0$ then the lower bound will be worse.

Branching s*trategy 2:* All three branches $X_{20,0} = 1$; $X_{20,8} = 1$; $X_{20,0} = 0$ and $X_{20,8} = 0$ lead to feasible solutions. The lower bounds of these three branches are compared, and the route having the smallest lower bound is selected. The lower bounds are provided on the below tree. Accordingly when $X_{20,0} = 1$ then the lower bound is the lowest, hence it is the selected route.

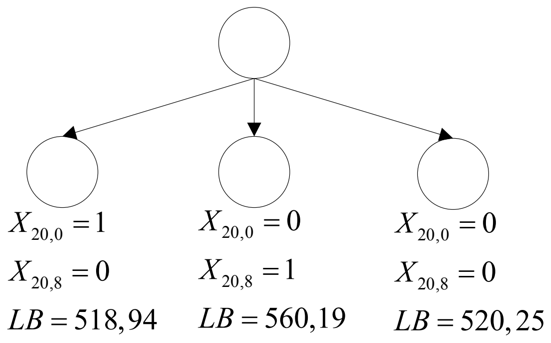

Figure 3-20 - A Part of Branch and Bound Tree for the Sample Project

3.3.4.2 Approximation Algorithm

We propose three heuristic procedures to find feasible approximate solutions to our problem. The first heuristic is LP based, runs in polynomial time and is used as an initial feasible solution in our branch and bound algorithm. The second heuristic is branch and bound based, runs in polynomial time and is proposed with the hope of obtaining high quality, fast solutions. The third heuristic is also branch and bound based, runs in exponential time and proposed with the hope of obtaining near optimal solutions.

LP Based Heuristic Algorithm

Our heuristic procedure to find an upper bound proceeds in two phases: construction and improvement.

Construction Phase

The construction part consists of 2 steps.

Step 1. The problem is first solved by relaxing the integrality constraints. Say t_{iLP} denotes the processing time of activity and c_{iLP} is its associated cost in the optimal relaxed solution.

Step 2. We decrease the cost of activity *i* and hence total cost of the project by setting activity *i* to the largest cost mode that is no larger than c_{iLP}. Our aim is to increase the completion time as small as possible by small increments. (From Property 14 we know that decreasing the cost of any activity does not violate feasibility.)

Consider the project whose associated information is given in Table 3-6. In the construction phase, the following mode assignments are made and the associated upper bound is found as 548. In the optimal LPR solution, variables associated with activity 17, 18 and 20 take fractional values as seen in Table 3-9.

For activity 17, $c_{17LP} = 588.35$ and the mode with maximum cost that is not larger than 588.35 is mode 5, hence we set $X_{17,5} = 1$.

For activity 18, $c_{18LP} = 1100.46$, and the mode with minimum duration that is not larger than 1100.46 is mode 6, hence we set $X_{18,6} = 1$.

For activity 20, $c_{20LP} = 487.19$, and the mode with minimum duration that is not larger than 487.19 is mode 4, hence we set $X_{20,4} = 1$.

The solution with the above assignments becomes feasible for the original problem, and the objective function value is the upper bound for the total completion time of the project.

Improvement Phase:

The solution found by the construction phase is improved using two procedures in sequel.

First Improvement

The construction heuristic can be improved by decreasing the durations of some activities. Such decreases are possible, as in construction phase we increase the costs that would give room to some decreases. For each critical activity we check whether it can be moved to its next smaller duration mode without violating feasibility. Among the activities that would lead to feasible decreases, we select the one that causes maximum reduction in total completion time. We terminate the first improvement whenever any further increase in durations leads to an infeasible solution.

For the sample project, activity 8 and activity 18 can be moved to their next smaller duration mode without violating the feasibility. Activity 8 is assigned to mode 1. When it is assigned to mode 2, total completion time of the project is decreased to 544. On the other hand when activity 18 is assigned to mode 7, instead of mode 6 then the total completion time is 540. Since movement of activity 18 causes the maximum reduction, we select to move activity 18.

We terminate the first improvement step, as any further increase in durations leads to an infeasible solution.

Second Improvement

The first improved solution is further improved by pairwise interchanges. For all pairs on the same path, we check whether increasing one activity duration while decreasing the other one reduces the total completion time without violating the feasibility. Among the pairs that qualify, we select the interchange that decreases the completion time most. We terminate whenever all interchanges are either infeasible or cannot lead to a reduction in completion time.

Branch and Bound Based Approximation Algorithm I

In this section, we apply our branch and bound algorithm using branching strategy1, but with no backtracking. We terminate whenever we find all integer solutions, no feasible solution, or no promising solutions (lower bounds are no smaller than the best known upper bound). For the sake of completeness *Step 0* and *Step 1* are restated below, although they are the same as in optimization algorithm. In *step 2*, *routing strategy 2* defined in the previous section is used.

> *Step 0.* Solve the LPR problem at the root node. Use LP based heuristic algorithm to find an initial feasible solution. Let upper bound be the total completion time of the solution and B^* be its total cost value. Recall from Property 8 that LPR solution has at most two fractional variables for each activity.
>
> *Step 1.* The variable with the highest fractional value is chosen. Say X_{ij} is the most fractional variable.
>
> *Step 2.* There are two branches associated with the selected variable. These branches are $X_{ij} = 1$ and $X_{ij} = 0$ as shown in the following figure.

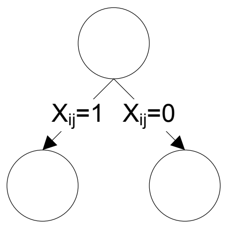

Figure 3-21 - Branching Strategy 1

Lower bound is used to find the route. LPR is solved for both branches. If both routes are feasible then the one with the smaller lower bound is followed. If both are infeasible then go to *Step 3*.

Step 3. Solve the LPR problem. There are three cases.

Case i. LP is infeasible, then stop. The solution giving the best upper bound is the solution of approximation algorithm.

Case ii. All variables in the optimal LPR solution are integer, the solution is optimal for the associated node. If $UB \geq LB$ (*LB* is the total completion time found in optimal LPR solution) then let $UB = LB$. Then stop. The solution giving the best upper bound is the solution of approximation algorithm.

Case iii. There are two cases.

a. LPR is feasible and fractional and $UB \geq LB$, then go to Step 1.
b. $UB < LB$, then stop. The solution giving the best upper bound is the solution of approximation algorithm.

In place of *Branching Strategy 1, Branching Strategy 2* could also be used to arrive at an approximate solution.

Branch and Bound Based Approximation Algorithm II

In this section we describe another approximation algorithm that is based on reducing the problem size and solving the reduced problem by branch and bound. The variables that take integer values in the optimal LPR solution are fixed and then branch and bound is applied for the fractional variables. The method consists of 3 steps.

Step 0. The LPR problem is solved.

Step 1. All variables that take integer values in the optimal LPR solution are fixed.

Step 2. Branch and bound algorithm is used to find the optimal values of the activities with continuous variables.

CHAPTER 4

COMPUTATIONAL RESULTS

In this chapter we design an experiment to test the performances of our solution algorithms together with reduction and bounding mechanisms. We first present our data generation scheme and state our performance measures. We then discuss the results of our preliminary experiment that are used to set the mechanisms of the main runs. Finally, we present our main experiment and discuss its results.

4.1 Data Generation

The data we use in our experiment are basically taken from Akkan et al. (2005). We use medium-sized problem instances to solve the budget problem and large-sized problem instances to solve the minimum deadline problem. We next discuss the problem parameters together with their settings.

In the project scheduling literature, Complexity Index (CI) and Coefficient of Network Complexity (CNC) are used to define the network complexity.

For an AoA representation CNC is the ratio of the number of activities to the number of events. Hence higher CNC results in higher number of activities.

CI is the other measure of complexity for the networks. For an AoA representation it provides a measure for the network's similarity to a series-parallel network, and computed as the minimum number of node duplications to convert the network into a series-parallel network. A higher CI either refers to higher number of precedence relations or more complex precedence relations that require dummy nodes and/or arcs. Hence higher the CI values associate to harder to solve networks. For the details of the index, we refer the reader to Bein et al. (1992).

The number of modes, $|m_i|$, is randomly determined from discrete uniform distribution. Two uniform distributions, $U[2,10]$ and $U[11,20]$, are used to see the effect of $|m_i|$ on the performance of our algorithms.

The durations and costs of the activity modes are discrete and generated as follows: The durations are generated from $U[3,123]$. Then they are sorted such that t_k is the *k*th smallest duration. The minimum cost, c_m, is generated from U[5,15]. Then c_{k-1} is set to $c_k + s_k(t_k - t_{k-1})$ where $s_{k-1} \in U[s_k, s_k + 3]$ or $s_{k-1} \in U[Max(1, s_k - 3), s_k]$

Three different time/cost functions are used: Concave, convex and hybrid. The types of time/cost functions affect the computational performance. For example, the linear programs are solved only for two modes per activity when the time cost function is concave (See Properties 3 and 10).

We try four different budget values: b_{max}, $b_{l\,arg\,e}$, $b_{average}$ and b_{small}. b_{min} and b_{max} are the minimum and maximum total cost values of all efficient solutions respectively. Using these values, we find $b_{l\,arg\,e}$, $b_{average}$ and b_{small} as follows:

$$b_{average} = \frac{b_{max} + b_{min}}{2}$$

$$b_{l\,arg\,e} = \frac{b_{max} + b_{average}}{2}$$

$$b_{small} = \frac{b_{min} + b_{average}}{2}$$

We consider 12 combinations with the following parameters for . $b_{l\,arg\,e}$, b_{mean} and b_{small}.

Table 4-1 – The Budget Problem Parameters

CI	CNC	Cost Function	Number of activities	Number of modes/activity
0	2	Concave	[29,30]	[1,10]
0	2	Convex	[29,30]	[1,10]
0	2	Hybrid	[29,30]	[1,10]
0	2	Concave	[29,30]	[11,20]
0	2	Convex	[29,30]	[11,20]
0	2	Hybrid	[29,30]	[11,20]
4	2	Concave	[34,35]	[1,10]
4	2	Convex	[34,35]	[1,10]
4	2	Hybrid	[34,35]	[1,10]
4	2	Concave	[34,35]	[11,20]
4	2	Convex	[34,35]	[11,20]
4	2	Hybrid	[34,35]	[11,20]

We solve each 12 combination by 4 different b values. Hence we use $12*4=48$ combinations. For each combination we try 10 problem instances. Therefore 480 problems are solved. We treat the problem with b_{max} value as a minimum deadline problem, as well.

For the minimum deadline problem, to see the performances of the algorithms on large-sized problem instances, we perform an additional experiment using the parameters shown in Table 4-2.

Table 4-2 – The Parameters for the Large-Sized Instances

CI	CNC	Cost Function	Number of activities	Number of modes/activity
13	5	Concave	85	[1,10]
13	5	Convex	85	[1,10]
13	5	Hybrid	85	[1,10]
13	5	Concave	85	[11,20]
13	5	Convex	85	[11,20]
13	5	Hybrid	85	[11,20]

For each of these 6 combinations, we solve 5 problems. We set a termination limit of 2 hours for all algorithms. We report the best solution found so far, when we stop at the termination limit.

4.1 Performance Measures

To evaluate the efficiency of our branch and bound algorithm we use the following performance measures.

1. Average and maximum CPU time in seconds
2. Average and maximum number of nodes generated
3. Number of unsolved instances
4. Average and maximum number of nodes generated to reach the optimal solution

The following measures are used to evaluate the performance of our heuristics.

1. Average and maximum CPU time
2. Average and maximum percentage deviation from the optimal solution

We evaluate the root node performances of the lower bounds by their percentage deviation from the optimal solution in average and maximum terms.

The algorithms are coded in Microsoft C# programming language and run on Microsoft Windows Operating System. The optimal LP solutions are found by Cplex 10.1.

4.2 Preliminary Experiments

In designing a preliminary experiment, we aim to see the effects of our mechanisms and continue the main run with the most effective components.

We use 30 instances whose parameters are tabulated in Table 4-3 to test the performance of mode elimination rules, branching strategies and heuristic procedures. Moreover the effect of the LPR bound on the performance of the branch and bound is investigated. For each of these three combinations, we solve 10 problem instances.

Table 4-3 – The Properties of Instances Used in Preliminary Experiments

Cost Function	Number of activities	Number of modes per activity
Concave	[29,30]	[1,10]
Convex	[29,30]	[1,10]
Hybrid	[29,30]	[1,10]

Upper Bound Selection for a Feasible Solution

We check the performance of the LPR based heuristic procedure when used as an initial feasible solution in a branch and bound algorithm and in a specified number of branching nodes with the hope of improving the best known feasible solution.

Table 4-4 gives the combinations used to test the upper bound effects. Table 4-5 and Table 4-6 report the associated results for the average CPU times and number of nodes, respectively.

Table 4-4 – The Combinations Used for UB Settings

	UB1	UB2	UB3
LPR Based Procedure at the Root Node	Yes	Yes	Yes
Applying UB Procedure at First 50 Nodes	Yes	No	-
Applying UB Procedure at First 100Nodes	-	No	Yes

We set the number of nodes for which the upper bound is computed to either 50 or 100 and see whether computing upper bounds in more nodes improves the performance or not.

Table 4-5 – Average CPU Times Regarding UB Finding Procedures

	Large	Mean	Small	Overall
UB1	1050.57	245.60	139.40	478.52
UB2	1035.73	295.57	263.37	531.56
UB3	1048.60	247.93	259.47	518.67

Table 4-6 – Average Number of Nodes Regarding UB Finding Procedures

	Large	Mean	Small	Overall
UB1	28834.43	6618.00	6749.80	14067.41
UB2	30852.57	7963.20	6863.27	15226.34
UB3	28782.70	6569.17	6735.90	14029.26

.As can be observed from Table 4-5 and Table 4-6 for $b = b_{Large}$, UB2 is slightly better than UB1 and UB3, however in general UB1 results in smaller CPU times. UB1 and UB3 result in smaller number of nodes. As we observe that computing upper bounds for more than 50 nodes does not count for an improvement, we set the limit to 100 nodes. Otherwise we would compute upper bounds for more than 100 nodes and study the results.

Mode Elimination Algorithm Selection

We try six different ways of implementing our mode elimination rules. Mode elimination rules can be used at the root node and/or every node of the branch and bound tree. We consider the short and long mode elimination rules together and separated them from costly mode elimination rule which is considerably more time consuming. We tabulate the notation used for the mode eliminations in Table 4-7. According to the notation, *MR2* corresponds to the case where the short and long mode elimination rules are used only at root node and no other mode elimination is checked. All experiment is conducted under the same conditions by branching strategy 1 and routing strategy 2. We use the LPR based heuristic at the first 50 nodes of the branch and bound tree.

Table 4-7 – The Notation Used for Mode Reduction Procedures

		MR0	MR1	MR2	MR3	MR4	MR5
Mode Eliminations at Root Node	Short and Long Mode Elimination	No	No	Yes	Yes	Yes	Yes
	Costly Mode Elimination	No	Yes	No	Yes	Yes	Yes
Mode Eliminations at Branching Nodes	Short and Long Mode Elimination	No	No	No	No	Yes	Yes
	Costly Mode Elimination	No	No	No	No	Yes	No

Table 4-8 and Table 4-9 report the average CPU times and the number of nodes generated for different mode elimination implementations and budget values. The last column of the tables gives the averages over all b values.

Table 4-8 – Average CPU Times for Different Mode Elimination Implementations

	Large	Mean	Small	Overall
MR0	1186.53	228.73	249.30	554.86
MR1	1192.90	228.57	249.63	557.03
MR2	1016.03	228.23	247.30	497.19
MR3	927.17	229.13	127.80	428.03
MR4	1050.57	245.60	139.40	478.52
MR5	1051.23	244.67	264.77	520.22
Overall	1070.74	234.16	213.03	505.98

Table 4-9 – Average Number of Nodes for Diffferent Mode Elimination Implementations

	Large	Mean	Small	Overall
MR0	35131.30	6622.83	6840.83	16198.32
MR1	35136.93	6622.83	6840.83	16200.20
MR2	30068.93	6621.70	6811.87	14500.83
MR3	27571.50	6621.70	6811.87	13668.36
MR4	28834.43	6618.00	6749.80	14067.41
MR5	28870.43	6621.70	6811.87	14101.33
Grand Total	30935.59	6621.46	6811.18	14789.41

As seen from the tables the best performance is due to *MR3*. That is when all rules are used at the root node but not at any branching node, the average CPU time is the smallest. Applying the mode elimination algorithms at each node of the branch and bound tree does not decrease the average CPU times. Though *MR4* leads to a higher number of mode eliminations than *MR3;* the effort spent for checking additional eliminations outweighs the additional savings. Thus we choose to apply mode elimination rules only at root node in our main runs.

Selection of Branching Strategy

We now test the effects of our branching rules on the performance of the branch and bound algorithm. We test two cases: An algorithm using *Branching Strategy 1* (BS1) and an algorithm using *Branching Strategy 2* (BS2). Both cases are checked under the same conditions with mode eliminations at all nodes, upper bound at the first 50 nodes.

The average CPU time and number of nodes results are tabulated in Table 4-10 and Table 4-11, respectively.

Table 4-10 – Average CPU Times Regarding Branching Strategies

Budget Value	Large	Mean	Small	Overall
BS1	1341.57	231.40	122.40	565.12
BS2	1196.07	238.50	130.90	521.82

Table 4-11 – Average Number of Nodes Regarding Branching Strategies

Budget Value	Large	Mean	Small	Overall
BS1	28834.43	6618.00	6749.80	14067.41
BS2	26571.73	4325.27	2292.03	11063.01

As can be observed from Table 4-10, when $b = b_{Large}$, BS2 is significantly slower than BS1. BS2 always results in fewer nodes than BS1, however at an expense of additional CPU times. This is due to the fact that, at each node three LPRs are solved in BS2 while two LPRs are solved in BS1. We conduct our main rules by BS1 due to its better time performance.

Selection of Routing Strategy

The optimal LPR solution used to find the LB is expected to have a strong impact on the efficiency of branch and bound algorithm. Hence using LB to define the next visit is tested and this strategy is denoted as RS1. RS2 is the alternative routing strategy that selects the branch with value zero for the most fractional variable.

We test two routing strategies for different budget values, under the same conditions with branching strategy 1, and upper bounds by LPR based heuristic for the first 50 nodes.

Table 4-12 and Table 4-13 report the average CPU times and the number of nodes, respectively. Table 4-12 shows that RS1 runs significantly faster than RS2 for all b values. Note from the table that the average CPU time is 12.02 seconds when RS2 is used, and this time is reduces to 7.58 seconds with RS1. The observations are in same line for the number of nodes, as can be seen from Table 4-13. On average RS1 and RS2 produce 14067.41 and 20291.14 nodes respectively. There is an exception when $b = b_{small}$ which can be attributed to the randomness.

Table 4-12 – Average CPU Times for Routing Strategies

Budget Value	Large	Mean	Small	Overall
RS1	1050.57	245.60	139.40	478.52
RS2	1465.13	418.27	284.57	722.66

Table 4-13 – Average Number of Nodes for Routing Strategies

Budget Value	Large	Mean	Small	Overall
RS1	28834.43	6618.00	6749.80	14067.41
RS2	45511.03	10576.13	4786.27	20291.14

Comparison of the Minimum Deadline and the Budget Algorithms

Recall that, the minimum deadline problem is a special case of the budget problem with $b = b_{max}$. Hence all procedures developed for the budget problem directly apply to the minimum deadline problem. Recognizing the special structure of the minimum deadline problem we derive tighter elimination mechanisms and treat the budget in the objective function and the project completion as a constraint. As the constraint, $S_{N+1} = T_{Min}$ is much tighter than the constraint $B \le b_{max}$, we expect higher performance from the minimum deadline problem when compared with the budget problem with $B \le b_{max}$. Our preliminary tests, based on the following parameter combinations have verified our expectations.

Table 4-14 – The Parameters for Preliminary Test Instances

Cost Function	Number of activities	Number of modes/activity
Concave	[29,30]	[1,10]
Convex	[29,30]	[1,10]
Hybrid	[29,30]	[1,10]

Table 4-15 and Table 4-16 compare the average CPU times and the number of nodes of branch and bound algorithms for the minimum deadline problem and the budget problem with $B \leq b_{max}$.

Table 4-15 – Average CPU Times for the Budget and Deadline Problems

Model	**Concave**	**Convex**	**Hybrid**	**Overall**
Budget	745.20	3000.60	1598.70	1781.50
Deadline	10.30	351.30	67.30	142.97

Table 4-16 – Average Number of Nodes for the Budget and Deadline Problems

Model	**Concave**	**Convex**	**Hybrid**	**Overall**
Budget	20020.80	82540.00	44680.30	49080.37
Deadline	311.90	10798.60	23468.70	11526.40

As can be observed from the above tables, the branch and bound algorithm designed for the minimum deadline problem produce significantly fewer nodes in significantly smaller CPU times. On average, the minimum deadline algorithm produces 11526.4 nodes in 0.02 seconds whereas the budget algorithm produces 4080.37 nodes in 0.29 seconds. The differences in performances of two algorithms are more pronounced for concave and hybrid functions. After verifying our expectations with the experimental results, we perform our main runs for $B \leq b_{max}$ case using the algorithms designed specifically for the minimum deadline problem.

4.3 Main Experiments

In this section we first discuss the performance of the solution algorithms for the budget problem, and then report the results for the minimum deadline problem.

Performance of the Branch and Bound Algorithm

Based on the results of our preliminary runs, we use *Branching Strategy 1* to solve the budget problem, after applying mode eliminations at only root node. LP based heuristic is used to find an upper bound at the root node and first 50 branching nodes. The LP based routing strategy, *RS1*, is used in branch and bound.

In our main experiment, we investigate the effects of the budget values, time/cost function types, number of activities and number of modes per activity on the performance of the branch and bound algorithm.

4-17 below reports the performance measures of the branch and bound algorithm for b_{large}, b_{mean} and b_{small} values.

Recall that we terminate the execution of the branch and bound algorithm after 2 hours. Each unsolved instances contribute to the total CPU time by 7200 seconds. The number of nodes searched till termination limit is counted in average and maximum number of nodes.

Table 4-17 – The Performance Measures of Branch and Bound Algorithm for b Values

	Large	Mean	Small	Overall
Number of Instances	120	120	120	360
Average CPU Time	4384.16	3055.39	1254.46	2898.00
Maximum CPU Time	7200.00	7200.00	7200.00	7200.00
Average Number of Nodes	129359.83	85337.93	31707.23	82135.00
Maximum Number of Nodes	258162.00	262739.00	215861.00	262739.00
Average Number of Nodes till Optimality	38798.63	24423.38	13037.98	25419.99
Maximum Number of Nodes till Optimality	223980.00	178425.00	174283.00	223980.00
Number of Unsolved Instances	64	43	14	121
Average % LB Deviation	5.29	2.88	1.14	3.10
Maximum % LB Deviation	10.91	7.25	3.28	10.91

As can be observed from Table 4-17, the problem is solved significantly faster when $b = b_{small}$. The high CPU times for b_{large} and b_{mean} can be attributed to the relatively high number of unsolved instances. Note that 64 and 43 out 120 instances remain unsolved for b_{large} and b_{mean} respectively. This number is 14 when $b = b_{small}$. As can be observed from the table, the performances of the lower bound deviations at the root node are parallel with the performance of the branch and bound algorithm. When the lower bound deviations are smaller, the number of nodes and the CPU times are lower. Note from the table that these deviations are the (largest) smallest when b value is the smallest (largest) and the best (worst) branch and bound algorithm performance associates to smallest (largest) b values. Note that the average and maximum lower bound deviations are 3.1% and 10.91% respectively. This excellent performance of the lower bounds can be attributed to the special property of the optimal LPR that divides each activity to at most two modes.

We next investigate the effect of activity time/cost functions on the performance of the branch and bound algorithm. Table 4-18 reports the results for the concave, convex and hybrid time/cost functions.

We expect that our algorithm performs better for concave time/cost functions as all LP relaxations are solved for only two modes (see Property 10). CPU times for the instances with concave time/cost functions are solved smaller than those with convex time/cost functions; however note that the instances with hybrid time/cost functions can be solved faster than the others. The reason for the satisfactory behavior of the branch and bound algorithm for the hybrid functions can be explained by the high performance of the lower bounds. The worst performance associates to the convex time/cost functions which is due to the high number of unsolved instances. As can be seen from the table % LB deviation at the root node is relatively small for the convex cost functions; however the optimal solutions could not be found quicker, as the consecutive mode assignments (see Property 9) cannot differentiate between two branches.

Table 4-18 – The Performance of the Branch and Bound Algorithm for Different Time/cost Functions

Function Type	Concave	Convex	Hybrid	Overall
Number of activities	120	120	120	360
Average CPU Time	2246.63	4623.79	1823.59	2898.00
Maximum CPU Time	7200.00	7200.00	7200.00	7200.00
Average Number of Nodes	70564.40	121083.48	54757.13	82135.00
Maximum Number of Nodes	262739	217883	238778	262739
Average Number of Nodes till Optimality	28810.36	27192.71	20256.92	25419.99
Maximum Number of Nodes till Optimality	223980	174283	216418	223980
Number of Unsolved Instances	29	70	22	121
Average % LB Deviation	4.38	2.26	2.67	3.10
Maximum % LB Deviation	10.91	7.33	9.61	10.91

Note that number of nodes visited till the optimal solution does not differ significantly between different time/cost functions. By looking at overall average number of nodes till optimality, we can state that our algorithm finds the optimal in the first one-third of the search and then visits the rest of the branch and bound tree to verify optimality.

Table 4-19 below reports the performance of our branch and bound algorithm for different project sizes. In the table NAc1 and NAc2 denote instance sets with 29-30 and 34-35 activities respectively. M1 and M2, on the other hand, denote the instance sets with the number of modes having $U[2,10]$ and $U[11,20]$ distributions respectively.

Table 4 19 – The Performance of the Branch and Bound Algorithm for Different Project Sizes

	NAc1			NAc2			
	M1	M2	NAc1 Overall	M1	M2	NAc2 Overall	Overall
Number of Instances	90	90	180	90	90	180	360
Average CPU Time	494.88	3673.91	2084.39	2328.68	5094.54	3711.61	2898
Maximum CPU Time	7200	7200	7200	7200	7200	7200	7200
Average Number of Nodes	14517.78	104475.24	59496.51	67788.92	141758.06	104773.49	82135.00
Maximum Number of Nodes	209509	258162	258162	220871	262739	262739	262739
Average Number of Nodes till Optimality	5510.13	29456.02	17483.08	25978.12	40735.70	33356.91	25419.99
Maximum Number of Nodes till Optimality	130014	223980	223980	205323	198151	205323	223980
Number of Unsolved Instances	2	41	43	21	57	78	121
Average % LB Deviation	2.46	2.88	2.67	3.58	3.5	3.54	3.1
Maximum % LB Deviation	9.61	10.91	10.91	9.78	10.29	10.29	10.91

We expect that larger the size of the project, harder to solve is the problem. As seen from Table 4-19, all results are in the line with our expectations without any exception. For both NAc1 and NAc2, the CPU times and number of nodes are smaller for M1. As there are unsolved instances for all instance sets, maximum CPU time is 7200 seconds. NAc2 has higher CPU time than NAc1. Number of nodes is significantly smaller for M1. Similarly, NAc2 has higher number of nodes as there are more decisions in the problem. The number of unsolved instances also supports that as the size of the problem gets larger, the problem is solved harder. %LB deviation presented in Table 4-19 states that the performance of the LPR is sensitive to the number of activities, but not to the number of modes. This is due to the fact that the optimal LPR assigns each activity to at most two fractional modes, independent from the number of modes (see Property 8).

Approximation Algorithms for the Budget Problem

In this section we analyze the performances of our branch and bound based heuristic (branch and bound with no backtracking), construction heuristic used in the first phase of the LPR based heuristic and upper bound produced by LPR based heuristic (used in the branch and bound algorithm) results for different *b* values, time cost functions and problem sizes. Branch and bound based heuristic is denoted as Heuristic I hereafter.

Table 4-20 reports the performance measures of the approximation algorithms for b_{large}, b_{mean} and b_{small}. The last column represents the results for all *b* values.

Table 4-, shows that Heuristic I runs in at most 3 seconds over all problem instances. The average CPU time by Heuristic I for b_{large} combination (1.32 seconds), is the largest when compared with other b values, however the differences are not significant. The deviations by Heuristic I is smallest for b_{small}, this is due to the better performance of the LPR based lower bounds for small b. The average deviation over all b values is 0.23 % and the maximum deviation is 2.84%. Hence Heuristic I should be favored if near optimal solutions are required in reasonable time.

Table 4-20 – The Performances of the Approximation Algorithms for different b-Values

b Value	Large	Mean	Small	Overall
Number of Instances	120	120	120	360
Average CPU Time of Heuristic I	1.32	1.08	0.91	1.10
Maximum CPU Time of Heuristic I	3.00	2.00	2.00	3.00
Average % Deviation of Heuristic I	0.39	0.19	0.09	0.23
Maximum % Deviation of Heuristic I	2.84	1.30	1.23	2.84
Average % Deviation of Construction	12.76	5.80	2.48	7.02
Maximum % Deviation of Construction	33.20	16.93	13.70	33.20
Average % Deviation of LPR Heuristic	1.43	0.75	0.42	0.87
Maximum % Deviation of LPR Heuristic	7.84	4.63	2.38	7.84

As seen in Table 4-20 the largest % deviations of the construction type heuristic are observed for b_{large}. On the other hand the smallest % deviation is due to b_{small}. This is due to the fact that for $b = b_{small}$, the feasible region gets smaller. Table 4-20 shows that for all b values the construction heuristic is improved significantly in the improvement phase and the maximum deviation is reduced from 33.20% to 7.84%. The maximum % deviation is due to the b_{large} case, which also the largest average % deviation. b_{small} has the smallest % deviation. 0.87% overall deviation from the optimal shows that branch and bound starts with an initial feasible solution which is very close to optimal. The CPU times of the construction and improvement phases were negligible, hence are not reported.

We next analyze the performance of the approximation algorithms for concave, convex and hybrid time/cost functions and report the results in Table 4-21.

We observe from Table 4-21 that the CPU times of heuristic I for different cost functions are close. The maximum CPU time which is due to the convex function differs from the minimum average CPU time which is due to the concave function, by only 0.20 seconds. The minimum average % deviation of heuristic 1 is 0.14% and is due to the convex functions. This contradicts with our expectations that convex time/cost functions are the hardest to solve ones. The explanation may be the following: For convex cost functions there are 70 unsolved instances, while for the others this number is less than 30. For the unsolved instances, in place of optimal solutions, we use the solution returned by the branch and bound at the limit. Hence the deviations for those instances are underestimates of the exact deviations.

The solution found in the construction phase deviates from the optimal solution about 7% on average, however the maximum deviations are not satisfactory (between 25% and 34%). After the improvement phase, for all time/cost functions the average deviations are reduced to 0.87%.

Table 4-21- Performance of Approximation Algorithms for Different Time/Cost Functions

Function	**Concave**	**Convex**	**Hybrid**	**Overall**
Number of Instances	120	120	120	360
Average CPU Time of Heuristic I	1.02	1.22	1.07	1.10
Maximum CPU Time of Heuristic I	2.00	3.00	2.00	3.00
Average % Deviation of Heuristic I	0.29	0.14	0.24	0.23
Maximum % Deviation of Heuristic I	2.84	2.47	2.06	2.84
Average % Deviation of Construction	7.23	7.19	6.63	7.02
Maximum % Deviation of Construction	30.41	33.20	25.40	33.20
Average % Deviation of LPR Heuristic	1.17	0.60	0.82	0.87
Maximum % Deviation of LPR Heuristic	7.84	4.32	4.40	7.84

Finally, we investigate the effect of problem size parameters, i.e., number of activities and number of modes, on the performances of the approximation algorithms. Recall that, the CPU times by the branch and bound algorithm increase significantly with an increase in the problem size. However, we do not expect a pronounced effect of the problem size on the CPU times of the approximation algorithms. This is because, the approximation algorithms solve only few LPRs and the LPs run in polynomial time. The results in Table 4-22 support our expectations. The difference between two problem sizes is less than one second. % deviation of heuristic 1 is less than 1% for all problem sizes and we do not observe any surprising result for the average deviations. The deviations for the construction phase are very small for small size instances. As can be seen from Table 4-22 , M1 and NAc1 consistently produce better quality solutions than M2 and NAc2 respectively. The performances of the improvement phase solutions deteriorate with an increase in the problem size, but not with the number of modes.

Table 4 22 – The Performance Measures of Approximation Algorithms for the Different Problem Sizes

	NAc1			Nac2			
Data	M1	M2	NAc1 Overall	M1	M2	Nac2 Overall	Overall
N	90	90	180	90	90	180	360
Average of Heuristic I CPU Time	0.83	1.18	1.01	1.04	1.36	1.20	1.10
Max of Heuristic I CPU Time	1.00	2.00	2.00	2.00	3.00	3.00	3.00
Average of % Deviation of Heur I	0.25	0.19	0.22	0.31	0.16	0.24	0.23
Max of % Deviation of Heur I	2.63	2.84	2.84	2.40	1.67	2.40	2.84
Average of % Deviation of Construction	4.69	5.93	5.31	8.26	9.20	8.73	7.02
Max of % Deviation of Construction	33.20	23.67	33.20	30.41	30.39	30.41	33.20
Average of % Deviation of UB at Root	0.77	0.71	0.74	0.90	1.10	1.00	0.87
Max of % Deviation of UB at Root	4.40	3.92	4.40	5.71	7.84	7.84	7.84

Optimal Solution Algorithms for the Minimum Deadline Problem

Based on the results of our preliminary experiments, we perform *Branching Strategy 1* and LP based routing strategy, *RS1*, in branch and bound algorithm for the minimum deadline problem. We use mode eliminations only at the root node. The LP based heuristic is used to find upper bounds at the root node and the first 50 branching nodes.

We observe that the performances of the minimum deadline problem are in line with those of the budget problem in most cases. The performance of the branch and bound algorithm for different time/cost functions and problem sizes are reported below.

Table 4-23 reports the performance of our branch and bound algorithm for concave, convex and hybrid time/cost functions.

Table 4-23 – The Performance of the Branch and Bound Algorithm for Different Time/cost Functions

Function Type	**Concave**	**Convex**	**Hybrid**	**Overall**
Number of Instances	40	40	40	120
Average CPU Time	225.80	1046.43	384.20	552.14
Maximum CPU Time	7200.00	7200.00	7200.00	7200.00
Average Number of Nodes	7181.70	31825.70	12240.10	17082.50
Maximum Number of Nodes	229726.00	230126.00	237107.00	237107.00
Average Node Number till Optimality	1802.50	6292.65	1019.75	3038.30
Maximum Node Number till Optimality	54871.00	89583.00	15672.00	89583.00
Average % LB Deviation	0.98	1.75	1.10	1.28
Maximum % LB Deviation	6.48	7.94	4.72	7.94
Number of Unsolved Instances	1	4	1	6

As can be observed from Table 4-23, the CPU times are smallest for the instances with concave time/cost functions. This is because, for concave functions, the LPRs are solved more efficiently with two modes for each activity. Note that the number of nodes generated by the convex and concave functions are close, but convex functions have significantly higher CPU times, as the time spent by the LPs at the nodes are much higher. The lower bound performance of the hybrid function is better than that of the convex function, with respective deviations of 1.1% and 1.75%. This difference in the lower bound performances at the root node justifies the better performance of the branch and bound algorithm for the hybrid functions over the convex functions.

Number of nodes visited until reaching the optimal is the highest for convex cost functions which is in line with our expectations. On average over all problem instances, the % LB deviations are below 1.3% and the maximum deviation is 8%. Small % LB deviations imply that we start the branch and bound algorithm with an LPR solution that is very close to the optimal solution. As we use LPR to find the branches to be visited, we reach the optimal solution at early nodes of the branch and bound tree. As can be observed from the table, on average, 17082.50 nodes are generated and the optimal solutions are found after generating 3038.30 nodes, i.e., about 80% of the effort is spent for proving the optimality of the solutions. We can conclude that truncated branch and bound algorithms that terminate at specified limits, are very likely to produce high quality solutions.

We next investigate the effects of the problem size parameters, i.e., the number of activities and number of modes, on the performance of the branch and bound algorithm. Since we observe that performance of the branch and bound algorithm is not affected from the interaction of these two parameters, we present the tables separately. In the tables, we use the following notation.

Table 4-24 - Notation Used for Number of Activities

	Number of Activities
NAc1	85
NAc2	102
NAc3	111/119
NAc4	128/135

Table 4-25 reports the performance of branch and bound for different number of activities.

We observe from the Table 4-25 that the CPU times are significantly high for NAc2, which is mainly caused by the high number of unsolved instances. However high number of unsolved instances for NAc2 cannot be explained by the number of activities but can be attributed to random effects. One can expect that the CPU times increase as the number of activities increases. However we cannot observe this in Table 4-25. The reason for this might be the reductions made by the mode elimination rules. As the problem size increases, more modes are eliminated by our mode elimination algorithms. Table 4-26 reports the average mode eliminations.

Table 4-25 – The Performance of the Branch and Bound Algorithm for Different Number of Activities

Instance Size	NAc1	NAc2	NAc3	NAc4	Overall
Number of Instances	30	30	30	30	120
Average CPU Time	188.17	1427.10	311.67	281.63	552.14
Maximum CPU Time	4724.00	7200.00	7200.00	7200.00	7200.00
Average Number of Nodes	5896.67	43874.07	10111.77	8447.50	17082.50
Maximum Number of Nodes	148392.00	230126.00	237107.00	214310.00	237107.00
Average Node Number till Optimality	2774.07	7965.00	745.00	669.13	3038.30
Maximum Node Number till Optimality	71658.00	89583.00	7697.00	15672.00	89583.00
Average % LB Deviation	1.06	1.85	1.18	1.02	1.28
Maximum % LB Deviation	3.89	7.94	5.52	3.00	7.94
Number of Unsolved Instances	0	4	1	1	6

Table 4-26 – The Number of Modes Eliminated for Changing Number of Activities

	NAc1	NAc2	NAc3	Nac4	Overall
Number of Modes Eliminated	684.07	798.03	990.43	1143.33	903.97

We observe that more modes are eliminated for the instances with large number of activities.

Table 4-27 reports that the number of nodes visited till finding the optimal solution does not follow a clear pattern. %LB deviations are the highest for NAc2 and smallest for NAc4. The satisfactory performance of the lower bounds leads to the small CPU time and number of nodes for NAc4.

Table 4-27 reports the performance of the branch and bound algorithm for the minimum deadline problem for two sets of the number of modes/activity. As stated in the previous section, M1 and M2 consist of the instances with the number of modes that follow discrete $U[2,10]$ and $U[11,20]$ distributions respectively.

We observe from Table 4-27 that the CPU times and the number of nodes are higher when there are more modes for each activity. This result is in line with our expectations. As can be seen from Table 4-28 , high number of mode eliminations cannot outperform the increasing effect of the number of modes.

Table 4-27 - Performance of the Branch and Bound Algorithm for Changing Number of Modes

	M1	**M2**	**Overall**
Number of Instances	60	60	120
Average CPU Time	493.35	610.93	552.14
Maximum CPU Time	7200.00	7200.00	7200.00
Average Number of Nodes	15024.50	19140.50	17082.50
Maximum Number of Nodes	229726.00	237107.00	237107.00
Average Node Number till Optimality	3557.60	2519.00	3038.30
Maximum Node Number till Optimality	89583.00	71658.00	89583.00
Average % LB Deviation	1.99	0.56	1.28
Maximum % LB Deviation	7.94	3.34	7.94
Number of Unsolved Instances	2	4	6

Table 4-28 – The Number of Modes Eliminated for Changing Number of Modes

	M1	**M2**	**Overall**
Number of Modes Eliminated	433.75	1374.18	903.97

The average number of nodes visited until finding the optimal solution is smaller for M2, due to its lower %LB deviation. %LB deviation is less than 3.34% for M2, while it is less than 7.94% for M1.

Approximation Algorithms for the Minimum Deadline Problem

We now investigate the performances of our LPR based initial feasible solution algorithm (both the construction phase and the improvement phase) and the branch and bound based approximation algorithm. The performances of the algorithms are reported for different activity time/cost functions and varying problem sizes. Table 4-29 reports the performance of the approximation algorithms for instances with concave, convex and hybrid time/cost functions.

Table 4-29 shows that heuristic 1 runs in 0.81 seconds on average and 6 seconds at worst. The resulting % deviations for all function types are consistently very low. The average and maximum deviations over all 120 instances are 0.31% and 2.5% respectively.

Table 4-29 – The Performance of Approximation Algorithms for DifferentTime/cost Functions

Function Type	**Concave**	**Convex**	**Hybrid**	**Overall**
Number of Instances	40	40	40	120
Average CPU Time for Heuristic 1	0.58	1.10	0.75	0.81
Maximum CPU Time for Heuristic 1	3.00	6.00	4.00	6.00
Average % Deviation of Heuristic I	0.22	0.38	0.33	0.31
Maximum % Deviation of Heuristic I	2.00	1.96	2.50	2.50
Average % Deviation of Construction	0.89	1.23	0.91	1.01
Maximum % Deviation of Construction	5.85	8.10	4.76	8.10
Average % Deviation of LPR Heuristic	0.68	1.03	0.85	0.85
Maximum % Deviation of LPR Heuristic	5.82	7.03	3.48	7.03

We did not report on the CPU times of the LPR based heuristics, as they were negligibly small. We find that the average % deviation of the construction phase of the LPR based heuristics is 1.01%. In the improvement phase the average deviation is reduced to 0.85%. No significant effect of the time/cost functions is observed.

We now study the effect of the number of activities on the performance of the algorithms and report the results in Table 4-30. We use the notation stated in Table 4-24.

Table 4-30 – The Performances of the Approximation Algorithms for Different Number of Activities

	NAc1	NAc2	NAc3	Nac4	Overall
Number of Instances	30	30	30	30	120
Average CPU Time for Heuristic 1	00.40	01.47	00.77	00.60	00.81
Maximum CPU Time for Heuristic 1	01.00	06.00	04.00	03.00	06.00
Average % Deviation of Heuristic I	0.30	0.37	0.29	0.27	0.31
Maximum % Deviation of Heuristic I	2.00	1.96	1.32	2.50	2.50
Average % Deviation of Construction	0.94	1.33	0.94	0.84	1.01
Maximum % Deviation of Construction	5.82	7.03	6.01	3.86	7.03
Average % Deviation of LPR Heuristic	0.93	1.21	0.70	0.57	0.85
Maximum % Deviation of LPR Heuristic	5.85	8.10	3.09	3.48	8.10

We observe from Table 4-30 that the CPU times and the deviations for all algorithms do not follow a pattern as the number of activities increases. For example NAc2 has the largest CPU time for Heuristic 1 and the largest deviations for the other algorithms.

Finally, we analyze the effects of the number of modes on the performance of the approximation algorithms and report the results in Table 4-30.

Table 4-30 shows that Heuristic 1 and the LPR based heuristic produce high quality solutions with an average overall deviation of 0.31 % and 1.01% respectively. The deviations are consistently small as can be verified from the low maximum overall deviations of 2.5% and 1.43%. As all deviations are very small, we could not observe any significant effect of the number of the modes on the performance of the algorithms.

Table 4-31 – The Performance of the Approximation Algorithms for Different Number of Modes

Data	**M1**	**M2**	**Overall**
Number of Instances	60	60	120
Average CPU Time for Heuristic 1	00.80	00.82	00.81
Maximum CPU Time for Heuristic 1	06.00	06.00	06.00
Average % Deviation of Heuristic I	0.52	0.10	0.31
Maximum % Deviation of Heuristic I	2.50	0.84	2.50
Average % Deviation of Construction	7.03	1.54	7.03
Maximum % Deviation of Construction	8.10	1.26	8.10
Average % Deviation of LPR Heuristic	1.67	0.35	1.01
Maximum % Deviation of LPR Heuristic	1.43	0.28	1.43

CHAPTER 5

CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

The Discrete Time/Cost Trade-off (DTCT) problem is an important research area in project management, particularly in view of the current emphasis on time-based competition (De et al., 1995). Of its obvious practical importance, the DTCT problems have attracted the attention of many researchers since early sixties.

In the project management literature, the DTCT problems are studied under two main categories: the deadline problem and the budget problem. Both problems are shown to be strongly NP-hard. The successive solutions of these problems are used to construct the time/cost trade-off curve.

Several procedures have been proposed for the deadline problem including bounding approaches and optimization procedures. However, despite its obvious practical importance, only little effort has been spent on the budget problem and to the best of our knowledge, there are two reported solution approaches which are based on dynamic programming. The dynamic programming approach is limited to solve only small sized problem instances. Hence more efficient optimization and approximation algorithms are needed. Recognizing this fact, we propose a branch and bound algorithm to solve moderate sized problem instances optimally and linear programming relaxation based heuristic procedures to solve large sized problem instances approximately. Our heuristic approaches run in polynomial time.

Our preliminary experiments to detect the effects of our reduction mechanisms and bounding approaches have revealed that the mode elimination mechanisms are very effective in reducing the problem size. The reduced problems can be efficiently solved by our branch and bound algorithm using the linear programming relaxation based lower bounds. We observe that in the absence of the linear programming relaxation based lower bounds, even the small sized problem instances could hardly be solved in reasonable times. The upper bounds, when applied to all nodes are not found to be effective. The effort spent to compute them outweighed the amount gained due to the additional node eliminations. Hence we used upper bounds only at a specified number of nodes (50 nodes), to find an initial feasible solution. Moreover the initial feasible solutions are found to be very close to the optimal solutions and this makes the frequent updates worthless.

Employing the results of our initial experiments, we perform an extensive computational study using large sized problem instances. Our aim is to detect the effects of the design parameters on the difficulty of the solutions. We found that the the number of activities, the number of modes and the project budget have strong influences on the problem difficulty. Moreover the types of the time/cost function used, are also effective such that the easiest problems associate to the concave functions as the associated linear programs are solved considering only two extreme modes for each activity.

The heuristics are found very effective, they produce close-to-optimal solutions in very small solution times. We suggest to use heuristic procedures in place of optimal solutions when the guarantee of optimality is not very essential.

We hope our study contributes to the current state of the art in discrete time/cost trade-off scheduling and our promising results stimulate opening new research areas. Two noteworthy extensions of our budget problem may be deriving more powerful mode elimination rules hopefully not based on the bounds, but exact values and investigating worst case complexity of our linear programming relaxation based upper bounds. Such a complexity result may be based on the maximum possible gap between the upper bound and the linear programming relaxation based lower bound.

In constructing the time/cost trade-off curve, the previous studies solve the deadline problem for all possible realizations of the project completion time. Our time/cost trade-off curve construction method is based on the successive solutions of the budget problem. Future research may consider some efficient ways of implementing our construction method. One alternative implementation might be to benefit from the optimal solution of the budget problem for a total cost value of b while solving the budget problem with a total cost value of b-1.

Our discrete time/cost trade-off problem assumes that there are no constraints imposed on the availability of the resources. When there are resources of limited quantity, the activities requiring the same resource (which is available for one unit at a time) cannot be processed simultaneously, shorter modes for some activities should be selected and/or the activity start times should be delayed to maintain resource feasibility. The modification of our procedures so as to include resource leveling and/or allocation decisions may fill another gap in the project management literature.

REFERENCES

Akkan, C., Drexl, A., Kimms, A. (2005). Network decomposition-based benchmark results for the discrete time/cost trade-off problem. *European Journal of Operational Research, 165,* 339-358.

Battersby, A. (1970). *Network Analysis for Planning and Scheduling*. New York: John Wiley & Sons, Inc..

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

Christian A. Demassey S. Neron E., (Eds.). (2008) *Resource-constrained project scheduling.* London: John Wiley & Sons, Inc..

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

Crowston, W. B. (1970). Decision CPM: Network Reduction and Solution. *Operational Research Quarterly, 21,* 435-452.

De, P., Dunne, E. J., Glosh, J. B., Wells, C. E. (1995). The discrete time/cost trade-off problem revisited. *European Journal of Operational Research, 81,* 225-238.

De, P., Dunne, E. J., Glosh, J. B., Wells, C. E. (1997). Complexity of discrete time/cost trade-off problem for project networks. *Operations Research, 45,* 302-306.

Demeulemeester, E., Retck, B. D., Foubert, B., Herroelen, W., Vanhoucke, M. (1998). New computational results on the dicrete time/cost trade-off problem in projet networks. *Journal of Operations Research Society, 49,* 1153-1163.

Elmaghraby S. E., (1977) *Activity Networks*. Newyork: John Wiley & Sons, Inc..

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

Fulkerson, D. R. (1961). A network flow computation for project cost curves. *Management Science, 7*, 167-178.

Hafızoğlu B. A. Discrete time cost trade-off problem in project scheduling. (Master Thesis, Middle East Technical University, 2007). E. Thesis.

Hindelang, T. J., Muth, J. F. (1979). A dynamic programming algorithm for decision CPM networks. *Operations Research, 27,* 225-241.

Kelley, J. E. (1961). Critical-path planning and scheduling: Mathematical Basis. *Operations Research, 3,* 296-320.

Meyer, W. L., Schafer, L. R., (1965). Extending CPM for multiform project time/cost curves. *Journal of Construction Division, 91,* 45-65.

Philips, S., Dessouky, M. I. (1977). Solving the project time/cost trade-off problem using the minimal cut concept. *Management Science, 24,* 393-400.

Philips, S. (1996). Project management duration/resource trade-off analysis: An application of the cut search approach. *The Journal of Operations Research Society, 47,* 697-701.

Robinson, D. R. (1975). Dynamic programming solution to cost-time trade-off for CPM. *Management Science, 22,* 158-166.

Skutella, M. (1998). Approximation algorithms for the discrete time/cost trade-off problem. *Mathematics of Operations Research, 23,* 909-929.

Vanhoucke, M., Debels, D. (2005). The discrete time/cost trade-off problem under various Assumptions and Heuristic Procedures. *Working Paper: Universiteit Gent*

Wysocki R. K., Beck R. J., Crane D. B. (2000). *Effective Project Management.* Newyork: New York: John Wiley & Sons, Inc..