A MULTI-MODE PROJECT SCHEDULING PROBLEM WITH A SINGLE NONRENEWABLE RESOURCE

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

BY

CANSU ALTINTAŞ

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

AUGUST 2017

Approval of the thesis:

A MULTI-MODE PROJECT SCHEDULING PROBLEM WITH A SINGLE NONRENEWABLE RESOURCE

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

Prof. Dr. Gülbin Dural Ünver
Dean, Graduate School of **Natural and Applied Sciences** _______________

Prof. Dr. Yasemin Serin
Head of Department, **Industrial Engineering** _______________

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

Examining Committee Members:

Assoc. Prof. Dr. Canan Sepil
Industrial Engineering Dept., METU _______________

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

Assist. Prof. Dr. Sakine Batun
Industrial Engineering Dept., METU _______________

Assist. Prof. Dr. Bahar Çavdar
Industrial Engineering Dept., METU _______________

Assoc. Prof. Dr. Öncü Hazır
Business Administration Dept., TED University _______________

Date: August 18, 2017

ABSTRACT

A MULTI-MODE PROJECT SCHEDULING PROBLEM WITH A SINGLE NONRENEWABLE RESOURCE

Altıntaş, Cansu
M.S., Department of Industrial Engineering
Supervisor: Prof. Dr. Meral Azizoğlu

August 2017, 74 pages

In this thesis, we consider a multi-mode project scheduling problem with a single nonrenewable resource. We assume that the resource is released in pre-specified times at pre-specified quantities and the resource is consumed at activity completions. The activities can be processed at different modes where a mode is defined by a processing time and a resource requirement amount. Our problem is to select the modes and timings of the activities so as to minimize the project completion time.

We develop a mixed integer linear model and present a branch and bound algorithm. The results of our experiments have revealed that the mathematical model can handle only small-sized problem instances with up to 20 tasks and branch and bound algorithm can solve problem instances with up to 100 tasks for some resource release profiles.

Keywords: Project scheduling, Multiple modes, Nonrenewable Resource, Branch and Bound Algorithm

ÖZ

TEK YENİLENEMEYEN KAYNAK İLE ÇOKLU MODLU PROJE ÇİZELGELEME PROBLEMİ

Altıntaş, Cansu
Yüksek Lisans Endüstri Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Meral Azizoğlu
Ağustos 2017, 74 sayfa

Bu tezde, çoklu modlu, tek yenilenemeyen kaynaklı, bir proje çizelgeleme problemini ele aldık. Kaynağın daha önce berlirlenen zamanlarda ve miktarlarda aktarıldığını ve aktivite bitiş zamanlarında harcandığını varsaydık. Aktiviteler değişik modlarda proses edilebilmekte ve modlar işlem süresi ve kaynak gereksinim miktarı üzerinden tanımlanmaktadır. Problemimiz, proje bitiş zamanını enazlayacak şekilde aktivite modlarının seçilmesidir.
Çalışmamızda, karmaşık tam sayılı doğrusal programlama modeli ve dal-sınır algoritması geliştirdik. Deneysel sonuçlarımız, matematiksek modelin iş sayısı 20'ye kadar olan küçük ölçekli problemleri çözdüğünü; dal-sınır algoritmasının ise bazı kaynak profillerinde boyutları 100 işe ulaşan büyük ölçekli problemleri çözebildiğini göstermiştir.

Anahtar Kelimeler: Proje çizelgeleme, Çoklu Modlar, Yenilenemeyen Kaynaklar, Matematiksel Model, Dal-Sınır Algoritması

To my dear Emre Eryiğit

ACKNOWLEDGMENTS

I am most grateful to my supervisor, Meral Azizoğlu, who has always been very positive, supportive and above all else, truly inspirational. Before we started our research, I was assigned to her course as a teaching assistant. So, I am initially grateful to this lucky assignment, that was how I could spend time with her in my first semester. The more I knew her, the more I adored her. From the first day of my graduate education, she guided me perfectly and encouraged me to pursue an academic career by being a great role model. Professionally, I will always be indebted to her for her great efforts and supports throughout our study. Personally, I will miss her a lot, wherever I go.

I would like to express my deep appreciation to my parents, Semra İlhan and Mustafa Altıntaş, for their unconditional love and support. They not only loved me but also allowed me to make my own decisions and gave me an ability to exercise free will.

I am thankful to my supervisor in TU/e, Zümbül Atan, for her support during my exchange semester in Eindhoven. I also want to thank Sakine Batun and Melih Çelik, for their valuable guidance on my doctoral application process and their efforts to organize the memorable end of year concert that I will always be remembering.

METU Industrial Engineering Department has been a wonderful place to study, work, learn, think and be inspired. I will always feel very fortunate to be a part of this community. I am deeply thankful to all members of my department.

Finally, I would like to express my gratitude to some friends who made my graduate journey a happy one. I had many enjoyable moments with Jeyhun Veliyev and Farid Taghiyev. I can never forget the victories we speak of. I feel their support, no matter how many miles they are away. I survived some of the toughest times of my life thanks to the support of my good old friends: Gözde

Çelebi, Ceren Karapekmez and Tuğçe Demircioğlu. Gözde always sent her little lucky wind, whenever I needed. Tuğçe was with me from my very first day to my very last day in Ankara. When I felt blues, Ceren was the cure with her lasting optimism. It gives me great peace and comfort to know that they will always be there for me.

Last but not least, thanks to Evrim Altıntaş for being the loveliest and most inspirational sister in the world. She always gives me the strength to achieve, and sometimes teaches me “not to be that stubborn”.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

A project is a set of interrelated activities to be executed over a fixed period to create a unique product or service. Project management has been an area that attracts the operational researchers for modelling and solving various problems in order to find effective and efficient ways for planning, organizing, directing, scheduling projects. Dating back to 1950s, problem natures, methodologies and solution approaches have been changing to adapt for the new practical problems.

As an important step of project management, project scheduling defines the start and finish times of the project activities. The earliest method in this area is the critical path method (CPM) that finds the minimum project completion time schedule for unlimited resource availability case. The CPM's assumption of unlimited availability of resources hardly holds in real life. Depending on the resource needs and resource types, various project scheduling problems can be defined. When the progress of the projects is subject to resource constraints, the associated problems are the resource constrained project scheduling problems (RCPSP). The basic resource categories are renewable and nonrenewable resources and new resource types have been introduced recently. Renewable resources are temporarily available on some time periods. They can be dedicated to some activities when they are available and they can be used again after the completion of these activities. Labor, machine, equipment are some notable examples for renewable resources. Nonrenewable resources are consumed with usage. The availability of a nonrenewable resource is limited with its release amount for the entire project.

It is common to assume that nonrenewable resources are released as a lump sum at the beginning of the project but they can be also released in a progressive way throughout the project. The project budget, simply money, can be a good example for nonrenewable resources. Recently, other resource types such as doubly constrained, partially renewable which have some features of renewable and nonrenewable resources are introduced.

A generalization of the RCPSP is the multi-mode RCPSP (MRCPSP). In the MRCPS problems each activity can be executed in one of the several modes where a mode is defined by a processing time and a resource requirement amount. Mode selection represents the trade-off in such a way that an activity can be performed with a shorter duration if higher amount of resource is consumed.

In this study, we assume that there is a single nonrenewable resource that is released in specified times at specified quantities in a progressive way. Resource units are consumed when the activities are completed. There are several activity execution modes, i.e., time/resource pairs for each non-dummy activity. Our decision is to select a mode for each activity to minimize the project completion time without violating resource constraints.

In the absence of mode decisions, our problem is solved in polynomial time by the Carlier and Rinnooy Kan’s (1982) algorithm. The algorithm is based on shifting the late start schedule of the CPM to maintain resource feasibility.

To the best of our knowledge, there exists a unique research by Azizoglu and et al. (2015) for the nonrenewable resource, mode selection problem. The study formulates the problem as a pure integer linear program and presents linear programming relaxation based heuristic approaches for its approximate solutions.

A practical situation that is in accordance with our problem can be exemplified from the construction industry. A client orders a building from the building contractor. The contractor has subcontractors to execute the project activities and pays them when any activity is completed. Subcontractors can choose to perform an activity with their own labor and equipment or outsource it. Depending on the selected execution way, that activity costs more and takes less time or vice versa.

Another practical situation is the software business where the customer demands a specific software application for their accounting operations from a software developing company. According to the bid signed between the customer and software company pre-specified amounts at pre-specified times should be paid by the customer upon the satisfactory delivery of the software sub-modules.

In this thesis, we first formulate the problem as a mixed integer programming model that finds the minimum project completion time schedule. We introduce lower and upper bounds on the optimal activity completion times and some mode elimination mechanisms in order to reduce the number of integer variables. We then present a branch and bound algorithm that enumerates the partial solutions based on the mode assignment decisions. Given the mode assignment decisions, the partial solutions are evaluated by using some relaxations and if possible, are eliminated using the properties of the optimal schedule. To the best of our knowledge, our branch and bound algorithm is the first optimization algorithm for the multi-mode, nonrenewable resource constrained project scheduling problem. We show that our branch and bound algorithm is much superior to the mathematical model that we propose and our mathematical model is much superior to the model proposed by Azizoğlu et al. (2015) in terms of number of integer decision variables.

The thesis is organized as follows: In Chapter 2, the comprehensive literature review of related problems is given. In Chapter 3, we give the problem definition and present the mathematical models: the previously presented pure integer programming model and our mixed integer programming model. We also discuss some special cases of our problem and develop some properties of the optimal solutions. Chapter 4 presents the branch and bound algorithm together with the lower bonding mechanisms. We report on the results of our preliminary and main experiments in Chapter 5. In Chapter 6, the concluding remarks and planned future studies are given.

CHAPTER 2

LITERATURE REVIEW

In this chapter, we first present one of the earliest, well-known algorithm as a review of the project scheduling problems in the absence of resource constraints and mode decisions. Then, we make a review of project scheduling problems with multiple activity execution modes where a mode of an activity is defined by its processing time and resource requirement amount. Then, we provide a review on resource constrained project scheduling problems with single and multiple modes.

2.1 Problems Without Resource Constraints and Mode Decisions

Single mode resource-unconstrained project scheduling problems can be solved in polynomial time by the well-known Critical Path Method (CPM) presented by Kelly and Walker (1959). The CPM is one of the earliest methods in this area that finds the schedule through the earliest and latest start and finish times for each activity without increasing the minimum project completion time. For the sake of completeness, we state the CPM. Since resource is not a concern, activity duration and precedence relations are the only parameters that define the problem. The following notation will be used throughout the thesis:

p_i: Processing time of activity i

IP_i: Set of immediate predecessors of activity i

IS_i: Set of immediate successors of activity i

Given these two parameters, we find the following decisions:

ES_i: Earliest start time of activity *i*

LF_i: Latest finish (completion) time of activity *i*

EF_i: Earliest finish time of activity *i*

LS_i: Latest start time of activity *i*

We define total slack time of an activity as the maximum amount of time an activity's completion time can be delayed without delaying the minimum project completion time.

TS_i: Total slack time of activity *i*

$TS_i = LS_i - ES_i = LF_i - EF_i$

When the total slack time of an activity is zero, it is a critical activity. If we increase the processing time of a critical activity, the project completion time directly increases. The activities with positive total slack values are non-critical activities. The critical path is a set with all critical activities.

CP: Set of critical activities

The CPM works as follows:

Initially, the earliest start times of the activities with no predecessors are set to 0. Using the precedence relations, we sum the earliest start times and processing times to find the earliest finish times. Earliest start time of an activity is the maximum of the early finish times of its predecessors. After all earliest start and finish times are calculated, the maximum of the earliest finish times is taken as the earliest project completion time. Then, we initialize latest project completion time with earliest project completion time. Latest completion time of an activity is the minimum start times of all its immediate successors. After, we have earliest and latest finish times, we calculate the total slack for each activity which is the difference between the earliest and latest finish times. When both times are equal and correspondingly, total slack is zero, the activity is critical. By finding total slack for each activity, we obtain the critical path. Stepwise demonstration of the algorithm is as follows:

```
Initialization:
$ES_i = 0$        $i : IP_i = \emptyset$
Main Body:
**Repeat**
    $ES_i = \underset{j \in IP_i}{Max}\{ES_j + p_j\}$        $i: \forall j \in IP_i$
    $ES_j$ is calculated
**Until** $ES_i$ for $i$=1, 2, ..., $N$ are calculated
$T = \underset{i}{Max}\{ES_i + p_i\}$
$LF_i = T$        $i : IS_i = \emptyset$
**Repeat**
    $LF_i = \underset{j \in IS_i}{Min}\{LF_j - p_j\}$        $i: \forall j \in IS_i$
    $LF_j$ is calculated
**Until** $LF_i$ for $i$=1, 2, ..., $N$ are calculated
Finalization:
$Total\ slack_i = LF_i - ES_i - p_i$        $i = 1, 2, ..., N$
$Critical\ activities = \{i = 1, 2, ..., N \mid Total\ slack_i = 0\}$
```

To illustrate the algorithm, we use the example project whose Activity on Node (AoN) representation is given in Figure 2.1 below. In AoN representation, each activity is represented as a node and arrows show the immediate precedence relations.

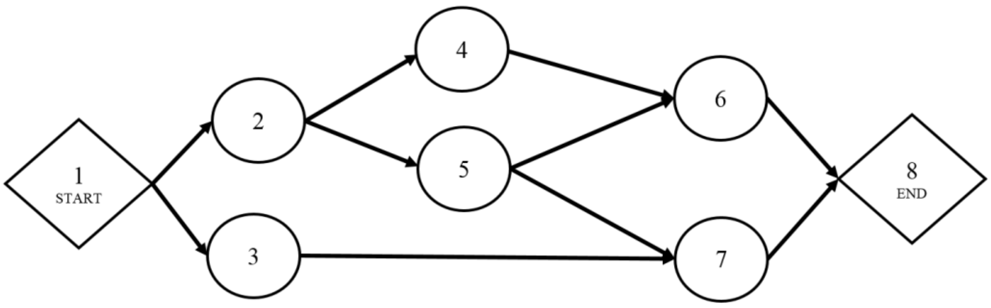

Figure 2.1 AoN Representation of the Example Project

In Figure 2.1, activities 1 and 8 are project start and project finish nodes, respectively. We define these dummy activities with zero processing times to denote the beginning and completion of the project. Table 2.1 below reports on the parameters of the network.

Table 2.1 Data of the Example Project

Activity	Immediate Predecessors	Duration (Hour)
1	-	0
2	1	2
3	1	3
4	2	4
5	2	3
6	4, 5	4
7	3,5	2
8	6, 7	0

We now give the step by step implementation of the CPM.

Initialization:

$IP_1 = \emptyset$; $ES_1 = 0$

Main Body:

$IP_2 = IP_3 = \{1\}$

$ES_2 = ES_3 = ES_1 + p_1 = 0 + 0 = 0$

We calculate ES_4 and ES_5 using ES_2:

$ES_4 = ES_2 + p_2 = 0 + 2 = 2$

$ES_5 = ES_2 + p_2 = 0 + 2 = 2$

Activities 6, 7, 8 have 2 immediate predecessors. Therefore,

$ES_6 = \text{Max}\ \{ES_4 + p_4; ES_5 + p_5\} = \text{Max}\ \{2 + 4; 2 + 3\} = 6$

$ES_7 = \text{Max}\ \{ES_3 + p_3; ES_5 + p_5\} = \text{Max}\ \{0 + 3; 2 + 3\} = 5$

$ES_8 = \text{Max}\ \{ES_6 + p_6; ES_7 + p_7\} = \text{Max}\ \{6 + 4; 5 + 2\} = 10$

We calculate the earliest finish time of each activity. Since activity 8 is the project end node with zero processing time, its completion time gives the entire project completion time.

$ES_8 = T = 10$

Using T, we now find latest finish times of activities.

$IS_8 = \text{Ø}; LF_8 = T = 10$

$IS_6 = IS_7 = \{8\}$ so $LF_6 = LF_7 = LF_8 – p_8 = 10 – 0 = 10$

$IS_5 = \{6, 7\}; LF_5 = \text{Min}\ \{LF_6 – p_6; LF_7 – p_7\} = \text{Min}\ \{10 – 4; 10 – 2\} = 6$

Activities 3 and 4 have only one immediate successor. Therefore,

$LF_4 = LF_6 – p_6 = 10 – 4 = 6;$

$LF_3 = LF_7 – p_7 = 10 – 2 = 8;$

$IS_2 = \{4, 5\}; LF_2 = \text{Min}\ \{LF_4 - p_4; LF_5 - p_5\} = \text{Min}\ \{6 - 4; 6 - 3\} = 2$

$IS_1 = \{2, 3\}; LF_1 = \text{Min}\ \{LF_2 - p_2; LF_3 - p_3\} = \text{Min}\ \{2 - 2; 8 - 3\} = 0$

Finalization:

Having calculated the earliest start and latest finish times of all activities, we find the total slack times. For example,

$TS_2 = LF_2 - ES_2 - p_2 = 2 - 0 - 2 = 0$

$TS_3 = LF_3 - ES_3 - p_3 = 8 - 0 - 3 = 5$

Hence, we know that activity 2 is a part of the path while activity 3 is not.

Table 2.2 summarizes the findings of the CPM implementation on example project.

Table 2.2 The CPM Results of Example Project

Activity	Immediate Predecessors	Duration (Hour)	Earliest Start Times	Latest Finish Times	Total Slack
1	-	0	0	0	0
2	1	2	0	2	0
3	1	3	0	8	5
4	2	4	2	6	0
5	2	3	2	6	1
6	4, 5	4	6	10	0
7	3,5	2	5	10	3
8	6, 7	0	10	10	0

The activities with zero total slack are the elements of the critical path set. The non-critical activities 3, 5 and 7 can be delayed as up to their total slack times. For instance, if activity 3 takes up to 4 more hours, the project does not delay. The thicker lines in Figure 2.2 illustrate the critical path. If any activity delays on critical path, the project completes later.

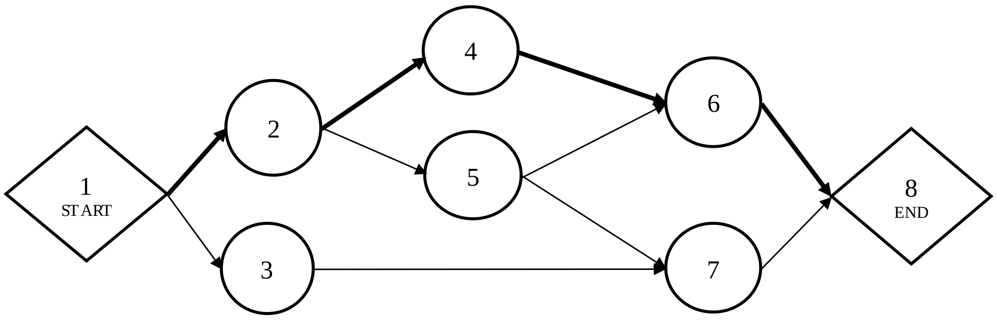

Figure 2.2 The Critical Path of the Example Project

2.2 Problems with Mode Decisions with no Resource Constraints

In project scheduling problems with multiple modes, each activity can be executed in one of the several modes where a mode is defined by a processing time and a resource requirement amount. The trade-off between time and resource consumption emerges with mode selection. For each activity, modes are non-dominated by each other. Depending on the selected mode, an activity is performed with a shorter duration and higher resource consumption or vice versa. In the literature, discrete time/cost trade-off problems (DTCTP) have discrete activity durations which are non-increasing functions of the single nonrenewable resource. Those problems have been studied under two categories: the deadline problem (DTCTP-D) and the budget problem (DTCTP-B).

The deadline problem minimizes the total resource cost subject to a specified deadline on the project completion time. Demeulemeester et al. (1998) present an exact branch and bound procedure to solve the deadline problem. Vanhoucke (2005), Akkan et al. (2005), Hafizoglu and Azizoglu (2010) are some of the most noteworthy studies on deadline problems.

The budget problem assumes a limited resource that is released at the beginning of the project as a lump sum and minimizes the project completion time. Since only a single resource is released at the beginning of the project, there is only one resource constraint. That is why, we present the literature review of the DTCTP here. The DTCTP-B is shown to be strongly *NP*-hard by De et al. (1997). For this problem, Skutella (1998) proposes heuristic solutions, Hazir et al. (2010) and Degirmenci and Azizoglu (2013) present optimization algorithms.

De et al. (1995) and Vanhoucke and Debels (2007) review the DTCTP and discuss their extensions.

2.3 Problems with Resource Constraints with no Mode Decisions

The assumption of unlimited availability of resources hardly holds in real life. When resource constraints are present, the problems are referred to as the resource constrained project scheduling problems (RCPSP). Even though new resource types have been introduced recently, Slowinski (1980) categorizes basic resource types as renewable, nonrenewable and doubly-constrained. *Renewable* resources are temporarily available on some time periods. They can be dedicated to some activities when they are available and they can be used again after the completion of these activities. Labor, machine, equipment can be examples for renewable resources. *Nonrenewable* resources are consumed with usage. The availability of a nonrenewable resource is limited with its release amount for the entire project. It is common to assume that nonrenewable resources are released as a lump sum at the beginning of the project but they can be also released in a progressive way throughout the project. The budget of the project can be an example for nonrenewable resources. The availability of *doubly-constrained* resource is limited both for the entire project and at every moment. These resources can be either consumed as nonrenewable resources or used during an activity execution as renewable resources. As shown by Talbot (1982), a pair of renewable and nonrenewable resources can replace a doubly-constrained resource.

Resource constrained project scheduling problems have many variants depending on the objective function used, randomness of problem parameters and many other assumptions. The following literature review is for the studies with project completion time minimization objective where parameters are assumed to be nonnegative and integer valued. Blazewicz et al. (1983) show that the RCPSP is an NP-hard problem.

Exact algorithms are proposed for the RCPSP under the presence of only renewable resources such as in the studies of Demeulemeester and Herroelen (1992), Brucker et al. (1998) and Mingozzi et al. (1998).

Carlier and Rinnooy Kan (1982) present an algorithm that finds optimal solution for the single nonrenewable resource with progressive resource arrival, project completion time problem. They present an exact method and show that this problem is solvable in polynomial time.

There are several reported exact solution approaches for the RCPSP with both renewable and nonrenewable resources such as the study of Chaleshtarti and Shadrokh (2014).

For hard problem instances that cannot be solved optimally, Kolisch and Hartmann (2005) summarize a large number of heuristics that have been proposed in the literature.

Since the RCPSP with their variants and extensions are abundantly studied, many survey and review studies are present in the literature. The most noteworthy of these studies are by Özdamar and Ulusoy (1995), Herroelen et al. (1998), Kolisch and Padman (2001), Hartmann and Briskorn (2010).

All the literature given so far are for the problems with deterministic problem elements. Herroelen and Leus (2005) provides a comprehensive review of the RCPSP under uncertainty.

2.4 Problems with Resource Constraints and Mode Decisions

A generalization of the RCPSP is the multi-mode RCPSP (MRCPSP). In the MRCPS problems each activity can be executed in one of the several modes.

Talbot (1982) introduces the mathematical model of the MRCPSP based on the RCPSP model.

Following the idea to enumerate partial schedules, exact approaches based on the Branch and Bound (B&B) method are presented by Kolisch et al. (1995), Sprecher et al. (1997), Sprecher and Drexl (1998). Hartmann and Drexl (1998) compare all available B&B methods presented so far and propose an alternative exact solution approach. Instead of using the B&B method, Zhu et al. (2006) use an exact branch and cut algorithm and present the integer linear programming (ILP) formulation of the problem. These studies assume the presence of both renewable and non-renewable resources with the objective of makespan minimization. Some noteworthy approximation studies are due to Jozefowska et al. (2001), Alcaraz et al. (2003) and Van Peteghem and Vanhoucke (2009).

Sabzehparvar and Seyed-Hosseini (2008) present an exact model for a MRCPSP in which the minimal or maximal time lags between a pair of activities vary depending on the selected modes and there are only renewable resources.

Azizoglu et al. (2015) study a MRCPSP with a single nonrenewable resource and prespecified resource releases throughout the project. They formulate the problem as a pure integer programming model and develop linear programming relaxation-based solution algorithms. Ozdamar and Ulusoy (1994) propose a heuristic approach named local constraint based analysis (LCBA) for the MRCPSP with a single nonrenewable resource. Weglarz et al. (2011) provides a comprehensive review on MRCPSP.

To the best of our knowledge, the most closely related study to ours is the study by Azizoglu et al. (2015). We study the same problem, propose an alternate mathematical programming model and an implicit enumeration technique, namely a branch and bound algorithm.

CHAPTER 3

PROBLEM DEFINITION

We study a multi-mode project scheduling problem with a single nonrenewable resource. We first define the problem and give its mixed integer linear programming model. We compare the model with the previously reported model in the literature. A condition for the existence of a feasible schedule and two special cases of the problem are stated. Finally the properties of the optimal solutions that help to reduce the problem size are discussed.

3.1 The Mathematical Model

There are N non-preemptive non-dummy activities. Two dummy activities 0 and N+1 represent the project start and completion, respectively. There exist precedence relations between activities. IP_i denotes the set of immediate predecessors of activity i. Activity i cannot start until all activities in IP_i are completed.

Activity i has m_i modes. Mode j of activity i is defined by two parameters as follows. p_{ij}, a processing time of activity i at mode j and c_{ij}, a resource requirement amount of activity i at mode j. Modes are assumed to be non-dominated, i.e., the longer the activity duration, the lower the resource consumption.

B_t denotes the amount of the resource released at time t and the resource is consumed at activity completions. If the resource released is not entirely used by the activities at some time unit, all remaining amount is transferred to the next time unit.

Our problem is studied by Azizoglu et.al. (2015). In our study, we reformulate the problem. For the sake of completeness, we present the mathematical model of

Azizoglu et al. (2015). Their decision variables x_{ijt} use time intervals of resource releases. The resource release is defined by two parameters, the amount B_t and the time instant I_t. I_t refers to the time instant at which the t^{th} payment is received. For instance, B_2 is the amount of resource released at time instant I_2 as illustrated in Figure 3.1 below.

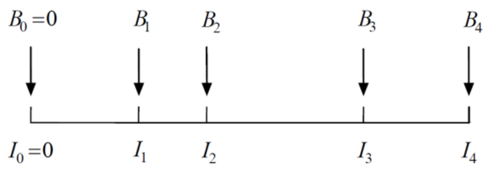

Figure 3.1 An Illustration of the Resource Releases

x_{ijt} denotes the mode selection and interval for each activity where;

$$x_{ijt} = \begin{cases} 1 & \text{if activity } i \text{ is completed on mode } j \text{ in time interval } (I_{t-1}, I_t\,] \\ 0 & \text{otherwise} \end{cases}$$

for $i = 1, 2,\ldots,N + 1$ and $j = 1, 2,\ldots, m_i$ and $t = 0, 1, 2,\ldots,T$

CT_i denotes the completion time of activity i ($i = 1, 2,\ldots,N+1$)

T is the number of time instants, i.e., number of resource releases.

So, $I_{t-1} + 1 \leq CT_i$ whenever $\sum_{j=1}^{m_i} x_{ijt} = 1$

The objective is to minimize the project completion time. They use the completion time of the dummy activity N+1 as below:

Minimize CT_{N+1}

Constraint set of the model is as follows:

$\sum_{j=1}^{mi} \sum_{t=0}^{T} x_{ijt} = 1$ $\qquad i = 1, 2,\ldots, N+1$ $\qquad$ (1)

$CT_i \geq CT_k + \sum_{j=1}^{mi} p_{ij} \sum_{t=1}^{T} x_{ijt}$ $\qquad i = 1, 2,\ldots, N+1;\ k \in IP_i$ $\qquad$ (2)

$$\sum_{i=1}^{N}\sum_{j=1}^{mi} c_{ij}\sum_{r=1}^{t} x_{ijr} \le \sum_{r=1}^{t} B_r \qquad t=0,1,2,\dots,T \tag{3}$$

$$CT_i \le \sum_{t=1}^{T} I_t \sum_{j=1}^{m_i} x_{ijt} \qquad i = 1, 2,\dots, N+1 \tag{4}$$

$$CT_i \ge \sum_{t=1}^{T}(I_{t-1} + 1) \sum_{j=1}^{m_i} x_{ijt} \qquad i = 1, 2,\dots, N+1 \tag{5}$$

$$CT_0 = 0 \tag{6}$$

$$CT_i \ge 0 \qquad i = 1, 2,\dots, N+1 \tag{7}$$

$$x_{ijt} \in \{0, 1\} \qquad i = 1, 2,\dots, N+1; j = 1, 2,\dots, m_i; t = 0, 1,\dots, T \tag{8}$$

Equation set (1) guarantees that each activity i is completed to exactly one mode and one time interval. Constraint sets (2) and (3) together satisfy the precedence relations and ensures that the total amount of resource released until time I_t is greater than or equal to the total amount spent till time I_t, respectively. Constraint sets (4) and (5) are to define the activity completion times. Equation (6) sets the completion time of activity 0 so the project starts at time zero. Constraint sets (7) and (8) are for non-negativity and for assignment variables, respectively.

In our model, we re-define the decision variables with the hope of having a more efficient model in terms of the computation time. As the number of indexes of a decision variable increases, the complexity of the model solutions increases. Likewise, integer decision variables are likely to increase the complexity more than the continuous decision variables. Recognizing this fact, we aim to define x_{ijt} as a continuous decision variable, and guarantee that it takes on value 0 or 1, at optimality. Recall that x_{ijt} involves both the mode assignment and timing decisions. We separate these two decisions via two sets of binary decision variables. In our notation, we use z_{ijt} instead of x_{ijt} and our idea is as stated below:

$$z_{ijt} = x_{it}{*}y_{ij}$$

where

$$x_{it} = \begin{cases} 1 & \text{if activity } i \text{ is completed at time t} \\ 0 & \text{otherwise} \end{cases}$$

for $i = 1, 2,\dots,N + 1$ and $t = 0, 1, 2,\dots,T_{UB}$

where T_{UB} is an upper bound on the optimal project completion time.

$$y_{ij} = \begin{cases} 1 & \text{if activity } i \text{ is completed on mode } j \\ 0 & \text{otherwise} \end{cases}$$

for $i = 1, 2,\ldots,N + 1$ and $j = 1, 2,\ldots,m_i$

$$z_{ijt} = \begin{cases} 1 & \text{if activity } i \text{ is completed on mode } j \text{ at time } t \\ 0 & \text{otherwise} \end{cases}$$

for $i = 1, 2,\ldots,N + 1$ and $j = 1, 2,\ldots, m_i$ and $t = 0, 1, 2,\ldots, T_{UB}$

Nonlinearity exists in the following definition of z_{ijt}

$$z_{ijt} = x_{it} {*} y_{ij}$$

We resolve this problem by using the following constraint set:

$$x_{it} + y_{ij} - 1 \leq z_{ijt}$$

for $i = 1, 2,\ldots, N+1$ and $j = 1, 2,\ldots, m_i$ and $t = 0, 1,\ldots, T_{UB}$

$$0 \leq z_{ijt} \leq 1$$

The constraint set above ensures that z_{ijt} takes values of 0 or 1. For example, if second mode is selected for activity 1 and this activity is completed at time 9, then

$y_{1,2} = 1$ and $x_{1,9} = 1$

Correspondingly, $z_{1,2,9}$ has to take value 1 as depicted below:

$$x_{1,9} + y_{1,2} - 1 \leq z_{1,2,9}$$

$$1 + 1 - 1 \leq z_{1,2,9}$$

Since $z_{1,2,9}$ is defined for the interval [0, 1], $z_{1,2,9}$ should be 1.

In all other cases where z_{ijt} cannot be 1, but can take infinitely many values in [0,1]. However, at optimality, it takes zero, due to our minimization concern.

As we redefine a decision variable, the number of variables and the number of constraints are affected. We now present our mathematical model.

Our objective is to minimize the project completion time and stated below:

Minimize $Z = \sum_{t=1}^{T} t\, x_{N+1,t}$

N+1 is the dummy activity to represent the project completion. So, when the objective function minimizes the task (activity) N+1's completion time, project completion time is minimized.

Our constraint set is given below:

$\sum_{t=0}^{T_{UB}} x_{it} = 1$ $\quad i = 1, 2,\ldots, N+1$ (1)

$\sum_{j=1}^{m_i} y_{ij} = 1$ $\quad i = 1, 2,\ldots, N+1$ (2)

$\sum_{t=0}^{T_{UB}} tx_{kt} + \sum_{j=1}^{m_i} p_{ij} y_{ij} \leq \sum_{t=0}^{T_{UB}} tx_{it}$ $\quad i = 1, 2,\ldots, N+1;\ k \in IP_i$ (3)

$\sum_{i=1}^{N} \sum_{j=1}^{mi} c_{ij} \sum_{r=1}^{t} z_{ijr} \leq \sum_{r=1}^{t} B_r$ $\quad t= 0, 1,\ldots,T_{UB}$ (4)

$x_{it}+ y_{ij}-1 \leq z_{ijt}$ $\quad i = 1, 2,\ldots, N+1;\ j = 1, 2,\ldots, m_i;\ t = 0, 1,\ldots, T_{UB}$ (5)

$0 \leq z_{ijt} \leq 1$ $\quad i = 1, 2,\ldots, N+1;\ j = 1, 2,\ldots, m_i;\ t = 0, 1,\ldots, T_{UB}$ (6)

$z_{000} = 0$ (7)

$x_{it} \in \{0, 1\}$ $\quad i = 1, 2,\ldots, N+1;\ t = 0, 1,\ldots, T_{UB}$ (8)

$y_{ij} \in \{0, 1\}$ $\quad i = 1, 2,\ldots, n+1;\ j = 1, 2,\ldots, m_i$ (9)

The model uses T_{UB} as an upper bound on the project completion time, and an upper bound on the optimal completion time of each activity. The computation of T_{UB} is discussed in Chapter 5.

Equation set (1) guarantees that each activity is completed exactly at one time unit and equation set (2) ensures that each activity is assigned to exactly one mode.

Constraint set (3) satisfies the precedence relationships among the activities and states that activity i cannot start until its predecessor activity k in IP_i is completed. Constraint set (4) ensures that for every time unit t, resource consumption of the completed activities cannot be greater than the total amount of resource released.

Constraint set (6) sets bounds on the continuous variables. Equation (7) ensures that the project starts at time *0*. Constraint sets (8) and (9) are for the assignment of variables x_{it} and y_{ij}, respectively.

Constraint set (5) constructs the relationship between binary decision variables x_{it}, y_{ij} and continuous decision variable z_{ijt} so that it ensures that the continuous variables take value 0 or 1. This is due to the structure of our objective function that penalizes the z_{ijt} values. For general objective functions, the following two sets of constraints should be included.

$z_{ijt} \leq x_{it}$ $\quad i = 1, 2,\ldots, N+1;\ j = 1, 2,\ldots,\ m_i;\ t = 0, 1,\ldots, T_{UB}$

$z_{ijt} \leq y_{ij}$ $\quad i = 1, 2,\ldots, N+1;\ j = 1, 2,\ldots,\ m_i;\ t = 0, 1,\ldots, T_{UB}$

3.2 Comparison with the Previously Reported Model

The model presented by Azizoglu et.al. (2015) (hereafter referred to as Model A) has two sets of decision variables: binary x_{ijt} and continuous CT_i. We define decision variables with two sets of two binary variables and one continuous decision variable with three indexes. Table 3.1 compares our model and the Model A in terms of the number of decision variables. In the table, N is the number of activities, M is an upper bound on the number of modes for any activity, T_{UB} is an upper bound on the optimal project completion time (we assume that there are T_{UB} time instants used by Model A)

Table 3.1 Number of Decision Variables in Two Models

	Number of Binary Variables	Number of Continuous Variables
Model A	N x M x T_{UB}	N
Our Model	N x M + N x T_{UB}	N x M x T_{UB}

The binary decision variables complicate the problem in a sharper way than the continuous variables. Note from the table that our model includes N x M + N x T_{UB}, i.e., $N(M + T_{UB})$ binary variables, whereas the existing model includes N x M x T_{UB}

binary variables. $M + T_{UB}$ is lower than M x T_{UB} when M and T_{UB} are higher than 1. The difference between $M + T_{UB}$ and M x T_{UB} increases significantly with increases in M and T_{UB}. Table 3.2 reports the number of decision variables for two problem instances: one with small sized with 20 tasks, 3 modes for each task and one with medium sized with 30 tasks, 4 modes for each task.

Table 3.2 Number of Decision Variables for an Example Project

	N = 20, M = 3, T_{UB} =155		N = 30, M = 4, T_{UB} =234	
	Binary Variables	Continuous Variables	Binary Variables	Continuous Variables
Model A	9300	20	28080	30
Our Model	3160	9300	7140	28080

Table 3.2 reveals that, as N increases T_{UB} increases. Note that the number of binary variables of the existing model is 9300 in instance N=20 and 28080 in instance N=30, whereas in our model, it is 3160 in instance N=20 and 7140 in instance N=30. Hence, the increase in the number of binary variables is faster in the existing model than our model.

3.3 A Feasibility Condition of the Mathematical Model

Our mathematical model returns a feasible solution if the following condition is satisfied:

$\sum_{i=1}^{n} c_{i1} \leq \sum_{r=1}^{T_L} B_r$

where c_{i1} is the lowest resource consumption for activity i and T_L is the time that the last resource unit is released. $\sum_{i=1}^{n} c_{i1}$ is the total amount of resource required when all activities are assigned to their first modes. This solution is the one having the minimum total resource requirement. If the minimum total resource requirement is no bigger than the total amount released, then the problem has a feasible solution. Otherwise, the resulting problem is infeasible.

3.4 Two Special Cases

We now state two special solvable in polynomial time cases of our problem that are defined in Azizoglu et al. (2015)

3.4.1 All Resource Units Are Released at Time Zero

When all resource units are released at time zero, the problem reduces to the discrete time/cost trade-off budget problem. Recall that the discrete time/cost trade-off budget problem is strongly *NP*-hard (De et al., 1997) so is our problem with scheduled resource releases.

3.4.2 Each Activity Has a Single Mode

When all activities have single mode, the problem reduces to the project completion time problem with a single nonrenewable resource that is solved by Carlier and Rinnooy Kan (1982). They introduce an exact algorithm and show that the problem is solvable in polynomial time.
For the sake of completeness, we state the algorithm below:

Algorithm Carlier and Rinnooy Kan (Algoritm C&R)

Step 1 Let p_i be time to perform activity i for i=1, 2,...,N

Find the latest activity completion times by the Critical Path Method (CPM).

Form a cumulative resource release graph by plotting t versus A_t, where A_t is the total amount of resource released till the end of the period t.

Form a cumulative resource requirement graph by plotting t versus , R_t where R_t is the total amount of resource required till the end of period t, when the activities complete at their latest completion times.

Step 2 Two cases arise:

Case A: $R_t \le A_t$ for all t (i.e., the cumulative resource requirement graph is beneath the cumulative resource release graph.)

The latest start schedule is optimal.

Case B: $R_t > A_t$ for at least one t

Find the optimal schedule by shifting the cumulative resource requirement graph (t versus R_t) to the right until all its R_t values lie on or below the A_t values of the supply graph.

An illustration of the C&R algorithm is presented on the example instance taken from Klastorin (2004) (see Figure 3.2).

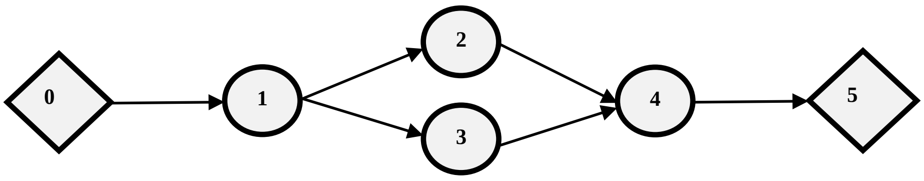

Figure 3.2 The Example Network

Table 3.3 gives the project data and the early and late activity completion times returned by the Critical Path Method (CPM). The critical path found by the CPM is 1–2–4 and has a completion time of 13 days.

Table 3.3 The CPM Results for The Example Network

Activity	Processing Times	Resource Requirement	Early Start Time	Late Completion Time
1	6	6	0	6
2	5	12	6	11
3	3	10	6	11
4	2	8	11	13

The early start times are the earliest start times of the activities while late completion times are the latest possible activity completion time without delaying the minimum project completion time. The resource arrivals are assumed to be at times 1, 7, 12, 20 with respective amounts of 5, 3, 10, and 18. The cumulative resource profiles are constructed as required (the dotted line, R_t) based on the late completion times of activities and as supplied (the solid line, A_t) as resource releases as in Figure 3.3. According to the algorithm, cumulative resource requirement line must be shifted to the right until it is on or below the cumulative resource supplied line. When minimum shift necessary is applied, project completion time of the problem is minimized.

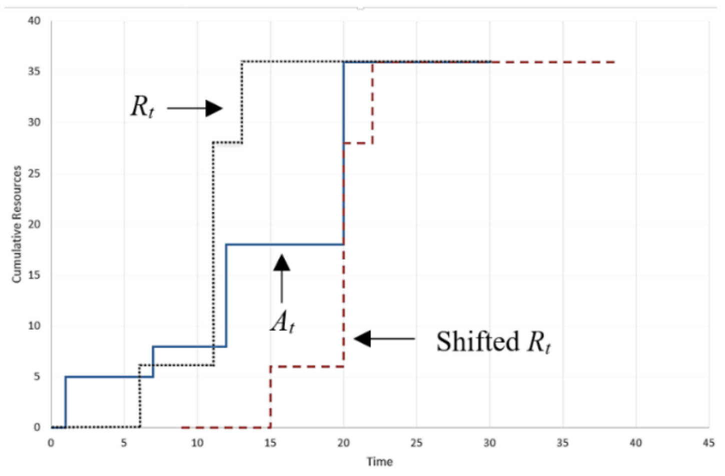

Figure 3.3 Graphical Illustration of the Algorithm for The Example Problem

3.5 Reductions in the Problem Size

Recall that the complexity of the mathematical model stems from the number of binary variables. The model resides T x N activity completion time variables, i.e., x_{it}s and $\sum_{i=1}^{n} m_i$ mode assignment variables, i.e., y_{ij}s. This follows, the complexity depends on the number of activities, number of activity modes and the magnitude of the activity completion times. In the following subchapters we propose some mechanisms for reducing the problem size by eliminating some of the decision variables.

3.5.1 Defining a Range for the Activity Completion Times

We try to reduce the number of binary variables by imposing lower and upper bounds on the activity completion times. Our aim is to define the activity completion times in a range, hence reduce the number of x_{it} variables used. In doing so, we let

$x_{it} \in \{0, 1\} \qquad i = 1, 2,\ldots,N+1; \;\; t = E_i,\ldots, \;L_i$

where E_i is a lower bound on the optimal completion time of activity i and L_i is an upper bound on the optimal completion time of activity i.

Finding E_i

In this chapter, we present lower bounds on the completion times of the activities en route to reducing the size of the model.

The Critical Path Method (CPM) finds the earliest and latest start and finish times for each activity in the absence of resource constraints. Since our decision variables are defined for the completion times, the earliest finish time (EF_i) from the CPM can be used by setting the processing times to their minimum possible values. The CPM parameters are then calculated.

EF_i: Earliest finish time of an activity i delivered by the CPM when the activity durations are taken from the modes with minimum durations.

The CPM assumes unlimited resource availabilities. In order to make bounds stronger, we modify the earliest finish times so as to include the resource availability. The modified earliest finish time is as stated below:

Note that c_{i1} is the minimum resource that can be consumed by activity i. P_i denotes the set of all, not only immediate, predecessors of activity i. The earliest time that $\sum_{k \in P_i} c_{k1} + c_{i1}$ units of resource becomes available gives another lower bound on the activity completion times. We let this time be ET_i.

Hence an overall lower bound on the completion time of activity i is E_i where

$E_i = \text{Max}\{EF_i , ET_i\}$

Modified earliest finish times, E_i values are used as to define the lower limits of the indices in our objective function and in constraint sets (1), (3), (4), (5) and (6).

Finding L_i

To find an upper bound on the optimal completion time of activity i, we use an upper bound on the optimal project completion time. We let T_{UB} be an upper bound on the optimal project completion time. To find T_{UB} we evaluate the following four promising feasible schedules.

i. Minimum activity time schedule

   For each activity i, use $p_{i\,1}$ and $c_{i\,1}$ where the first mode of an activity is the one having minimum processing time and using the maximum resource.

Apply Algorithm C&R to find the minimum project time schedule and let Z_1 be the objective function value of the resulting schedule.

ii. Minimum resource usage schedule

For each activity i, use $pi_{m(i)}$ and $c_{i\ m(i)}$ where the m_i^{th} mode of an activity is the one having maximum processing time and using the minimum resource.

Apply Algorithm C&R to find the minimum project time schedule and let Z_2 be the objective function value of the resulting schedule.

iii. Median activity time schedule(s)

Case 1. m_i is odd, set $k = m_i / 2$, round up k to the nearest integer

For each activity i, use $p_{i\ k}$ and $c_{i\ k}$

Case 2. m_i is even, set $k = m_i / 2$

For each activity i, use $p_{i\ k}$ and $c_{i\ k}$

Apply Algorithm C&R to find the minimum project time schedule and let Z_3 be the objective function value of the resulting schedule.

Case 3. m_i is odd, set $k = m_i / 2$, round down k to the nearest integer

For each activity i, use $p_{i\ k}$ and $c_{i\ k}$

Case 4. m_i is even, set $k = m_i / 2$

For each activity i, use $p_{i\ k}$ and $c_{i\ k}$

Apply Algorithm C&R to find the minimum project time schedule and let Z_4 be the objective function value of the resulting schedule.

Note that minimum activity time schedule may be promising as the activities are likely to be completed early due to their low processing times. Minimum resource usage schedule may be also promising as the activities are not likely to wait for the resource releases. The disadvantages of using minimum activity time and minimum resource usage schedules are their high resource and long processing requirements, respectively. Median time and median resource schedule(s) somewhat dispels those disadvantages by consuming lower resource than the minimum activity time schedule and taking shorter activity time than minimum resource usage schedule.

As an overall upper bound on the optimal project completion time, T_{UB}, we use $\min\{Z_1, Z_2, Z_3, Z_4\}$.

T_{UB} is used to define upper bounds on the activity completion times, UB_is as follows:

Apply CPM by setting $p_i = p_{i,1}$. Let T_{CPM} be the resulting project completion time and LC_i be the latest completion time of activity i. Note that LC_i is the latest time that activity i completes without exceeding the project completion time,

$LC_i + T_{UB} - T_{CPM}$ is the latest time that activity i completes without exceeding T_{UB}. As we are using the lowest activity times, $LC_i + T_{UB} - T_{CPM}$ is an upper bound on the optimal completion time of activity i.

Hence an upper bound on the completion time of activity i is L_i where $L_i = LC_i + T_{UB} - T_{CPM}$

3.5.2 Mode Eliminations

We now introduce the mode elimination rules for our makespan minimization problem. We aim to reduce the search size by eliminating some modes that cannot lead to an optimal or a feasible solution.

Long Mode Eliminations

We first state a theorem for the elimination of the long modes that would lead to non-promising solutions.

Consider an activity k performed at mode j with its original processing time and resource consumption, $p_{k,\,j}$ and $c_{k,\,j}$, respectively. For all other activities we use $p_{i\ 1}$ and $c_{i\ m(i)}$, $\forall\ i \neq k$ where 1st mode of an activity is the one with minimum processing times and last (m_i^{th}) mode of an activity is the one with minimum resource requirement. Let T be the project completion time when Algorithm C&R is applied to this single mode instance.

Through the following theorem we show that if $T > T_{UB}$ then activity k cannot be assigned to mode j in any optimal solution.

Theorem 1: If $T > T_{UB}$ then activity k cannot be assigned to mode j in any optimal solution.

Proof: Note that the single mode problem takes the activity times at smallest possible values and resource requirements at their lowest possible levels. This follows the resulting project completion time T is a lower bound on the optimal project completion time when activity i is assigned to mode j. If $T > T_{UB}$ where T_{UB} is the project completion time of any feasible solution, then activity i cannot be assigned to mode j in any optimal solution. □

Furthermore, if T_{UB} is the deadline of the project then in any feasible solution activity i cannot be assigned to mode j.

Using the result of the above Theorem 1, mode j of activity i is eliminated, however, not necessarily the modes of activity i with longer processing times. This is due to the fact that the longer activities may be processed earlier due to their lower resource requirements. In the Theorem 2 below, we will state the condition that we can eliminate more than one mode:

For activity k, we use $p_{i\,k}$ and $c_{k\ m(k)}$ and for each activity i, $p_{i\,1}$ and $c_{i\ m(i)}$, $\forall\ i \neq k$. Let T be the project completion time when Algorithm C&R is applied to this single mode instance. Through the following theorem we show that if $T > T_{UB}$ then activity k cannot be assigned to modes j through m_i in any optimal solution.

Theorem 2: If $T > T_{UB}$, then activity k cannot be assigned to modes j through m_i in any optimal solution.

The proof is omitted as it directly follows that of Theorem 1.

High Resource Mode Eliminations

We now state a theorem for the elimination of the high resource usage modes that would lead to infeasible solutions.

We let S_i be the set of activities whose latest completion times are earlier than the latest completion time of activity i.

Accordingly, $S_i = \{k \mid L_k \leq L_i\}$

Theorem 3: If $c_{i,j} + \sum_{k \in S_i} c_{k,m_k} > \sum_{t=1}^{L_i} B_t$ then activity i cannot be assigned to modes j, j-1, j-2,…1.

Proof: Note that S_i contains all activities that should finish before the latest completion time of activity i, L_i. Hence the total resource consumption by the activities in S_i and activity i should not exceed the total amount of resource consumed by the end of time L_i. This follows; once activity i is assigned to its r^{th} mode, the following constraint should hold.

$$c_{i,r} + \sum_{k \in S_i} \sum_j c_{k,j} X_{k,j} \leq \sum_{t=1}^{L_i} B_t$$

Note that $\sum_{k \in S_i} \sum_j c_{k,j} X_{k,j} \geq \sum_{k \in S_i} c_{k,m_k}$ if $c_{i,r} + \sum_{k \in S_i} c_{k,m_k} > \sum_{t=1}^{L_i} B_t$ then activity i cannot be assigned to mode r without violating feasibility.

If $c_{i,r} + \sum_{k\in S_i}\sum_j c_{k,j}X_{k,j} \leq \sum_{t=1}^{LC_i} R_t$ cannot be satisfied by mode r then it cannot be satisfied by any mode with higher resource consumption, i.e., the modes 1 through r should be eliminated. □

For the sake of completeness, we restate our mathematical model that uses the completion time bounds and mode elimination rules. Note that, m_i is modified if a mode of task i is eliminated due to our mode elimination mechanisms.

$$x_{it} = \begin{cases} 1 & \text{if activity } i \text{ is completed at time } t \\ 0 & \text{otherwise} \end{cases}$$

for $i = 1, 2,\ldots,N + 1$ and $t = E_i, E_i+1,\ldots, L_i$

$$y_{ij} = \begin{cases} 1 & \text{if activity } i \text{ is completed on mode } j \\ 0 & \text{otherwise} \end{cases}$$

for $i = 1, 2,\ldots,N + 1$ and $j = 1, 2,\ldots,m_i$

$$z_{ijt} = \begin{cases} 1 & \text{if activity } i \text{ is completed on mode } j \text{ at time } t \\ 0 & \text{otherwise} \end{cases}$$

for $i = 1, 2,\ldots,N + 1$ and $j = 1, 2,\ldots, m_i$ and $t = E_i, E_i+1,\ldots, L_i$

Minimize $Z = \sum_{t=1}^{T} t\, x_{N+1,t}$

Subject to

$\sum_{t=E_i}^{Li} x_{it} = 1$ $\quad i = 1, 2,\ldots, N+1$ (1)

$\sum_{j=1}^{m_i} y_{ij} = 1$ $\quad i = 1, 2,\ldots, N+1$ (2)

$\sum_{t=E_k}^{L_k} t x_{kt} + \sum_{j=1}^{m_i} p_{ij} y_{ij} \leq \sum_{t=E_i}^{L_i} t x_{it}$ $\quad i = 1, 2,\ldots, N+1;\ k \in IP_i$ (3)

$\sum_{i=1}^{N}\sum_{j=1}^{mi} c_{ij} \sum_{r=E_i}^{\min\{t,L_i\}} z_{ijr} \leq \sum_{r=0}^{t} B_r$ $\quad t = 0, 1,\ldots, T_{UB}$ (4)

$x_{it} + y_{ij} - 1 \leq z_{ijt}$ $\quad i = 1, 2,\ldots, N+1;\ j = 1, 2,\ldots, m_i;\ t = E_i, E_i+1,\ldots, L_i$ (5)

$0 \leq z_{ijt} \leq 1$ $\quad i = 1, 2,\ldots, N+1;\ j = 1, 2,\ldots, m_i;\ t = E_i, E_i+1,\ldots, L_i$ (6)

$z_{000} = 0$ (7)

$x_{it} \in \{0, 1\}$ $\quad$ $i = 1, 2,\ldots, N+1;\ t = E_i, E_i+1,\ldots, L_i$ $\quad$ (8)

$y_{ij} \in \{0, 1\}$ $\quad$ $i = 1, 2,\ldots, n+1;\ j = 1, 2,\ldots, m_i$ $\quad$ (9)

To see the effect of the problem size reduction mechanisms on the number of decision variables, we take an example instance from the project network with 20 tasks. Consider a single task, Task 11 which has initially 4 modes. In our updated model, one of the task modes is eliminated by the mode elimination mechanisms.

Table 3.4 The Changes in The Number of Decision Variables

	The Lower Bound	The Upper Bound	Number of Modes	Number of $x_{11,t}$	Number of $z_{11,j,t}$
Our Model without Reduction	0	114	4	114	456
Our Model with Reduction	11	86	3	75	225

T_{UB} for this particular instance is 114 time units. Table 3.4 shows the changes in the lower and upper bound on the completion time of Task 11.

So, the number of integer decision variables decreases from 114 to 75 and the number of continuous decision variables decreases to 225 from 456, when the bounds and mode elimination mechanisms are involved. Note that, these decreases are only for a single task of a small-sized project. The improvements due to these mechanisms would be much more significant when all tasks and larger-sized projects are considered.

CHAPTER 4

THE BRANCH AND BOUND ALGORITHM

The problem of minimizing project completion with mode decisions and arbitrary resource release times reduces to the discrete time/cost deadline problem, when all resource units are released at time zero. The discrete time/cost deadline problem is NP-hard in the strong sense (see De et al. (1997)), so is our problem with arbitrary resource release times. This complexity result justifies the use of an implicit enumeration technique. We present a branch and bound algorithm to find the optimal mode assignments and the optimal project completion time.
We first present our branching scheme.

We start with the first activity, that is the dummy activity that represents for the project start. At each level of the branch and bound tree, we select a task. At level *i*, we consider task *i* and generate m_i nodes: each representing the assignment of task *i* to one of the modes.

Figure 4.1 illustrates our branching scheme:

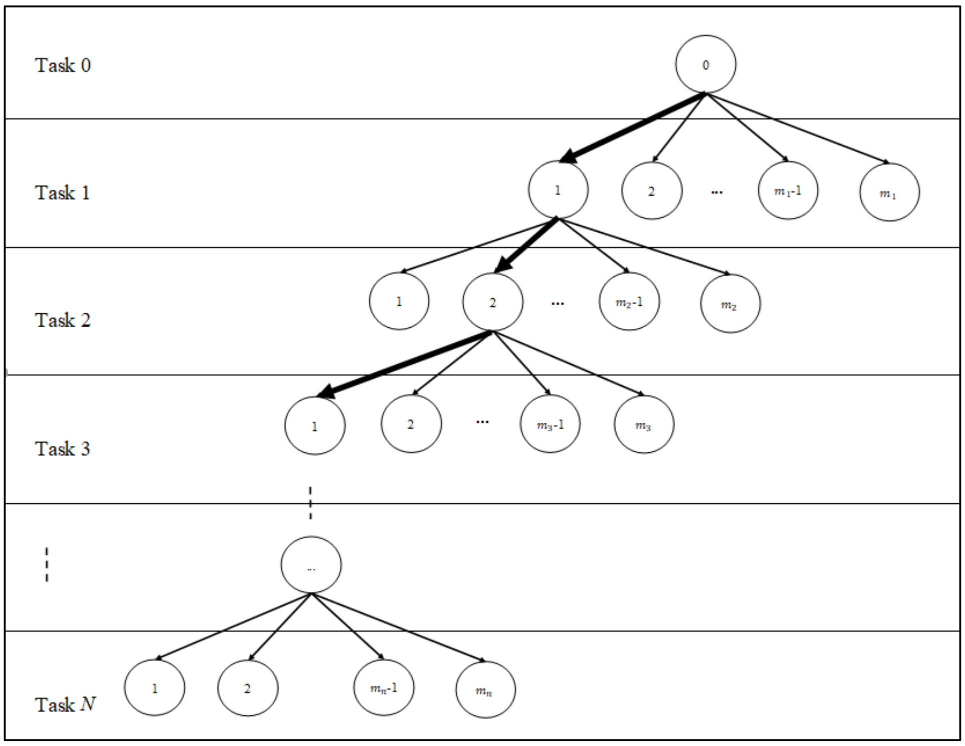

Figure 4.1 The BAB Branching Scheme

A node at the BAB tree represents a partial solution with a set of tasks with assigned modes. For example, in the BAB tree depicted in Figure 4.1, the shaded selection represents the assignment of task 1, 2 and 3 to modes 1, 2 and 1, respectively. At each node at level *i*, m_{i+1} nodes are emanating for representing an assignment of a mode to task *i*+1.

We implicitly evaluate m_1 partial solutions at level 1, m_1 x m_2 partial solutions at level 2 and so $\prod_{i=1}^{r} m_i$ partial solutions at level *i*. Hence, at the final level *N*, we implicitly enumerate all feasible $\sum_{k=1}^{N-1} \prod_{i=1}^{k} m_i$ partial solutions (nodes) and $\prod_{i=1}^{N} m_i$ complete solutions.

Our algorithm starts with an initial upper bound found through the following procedures that we used in Chapter 3 for problem size reductions. We let *UB* to denote the initial upper bound of the BAB algorithm.

To find *UB* we evaluate the following four promising feasible schedules.

i. Minimum activity time schedule

For each activity i, use $p_{i\,1}$ and $c_{i\,1}$ where the first mode of an activity is the one having minimum processing time and using the maximum resource.

Apply Algorithm C&R to find the minimum project time schedule and let Z_1 be the objective function value of the resulting schedule.

ii. Minimum resource usage schedule

For each activity i, use $p_{i\,m(i)}$ and $c_{i\,m(i)}$ where the m_i^{th} mode of an activity is the one having maximum processing time and using the minimum resource.

Apply Algorithm C&R to find the minimum project time schedule and let Z_2 be the objective function value of the resulting schedule.

iii. Median activity time schedule(s)

Case 1. m_i is odd, set $k = m_i / 2$, round up k to the nearest integer

For each activity i, use $p_{i\,k}$ and $c_{i\,k}$

Case 2. m_i is even, set $k = m_i / 2$

For each activity i, use $p_{i\,k}$ and $c_{i\,k}$

Apply Algorithm C&R to find the minimum project time schedule and let Z_3 be the objective function value of the resulting schedule.

Case 3. m_i is odd, set $k = m_i / 2$, round down k to the nearest integer

For each activity i, use $p_{i\,k}$ and $c_{i\,k}$

Case 4. m_i is even, set $k = m_i / 2$

For each activity i, use $p_{i\,k}$ and $c_{i\,k}$

Apply Algorithm C&R to find the minimum project time schedule and let Z_4 be the objective function value of the resulting schedule.

UB is the minimum objective function value of the feasible solutions above, i.e.,

$UB = \min\{Z_1, Z_2, Z_3, Z_4\}$.

We update the upper bound in two ways:

i. At some intermediate nodes, by assigning the smallest processing time mode to all unassigned tasks.
ii. At the final level, whenever a complete solution with a smaller project completion time is reached.

We eliminate the modes by using the results of our optimality properties, i.e., whenever the total minimum resource requirement is higher than the remaining total resource releases. Formally, we fathom the node representing the assignment of mode *j* to task *k*, if

$$c_{k,j} + \sum_{i \notin S}^{N} c_{i,1} \geq \sum_{t=1}^{T} B_t - \sum_{i \in S}^{k-1} c_{i,j} x_{i,j}$$

where S is the set of assigned tasks, i.e., $S = \{1, 2, \ldots, j-1\}$ and $c_{i,1}$ is the minimum resource consumption mode of activity *i*.

For each uneliminated mode, we calculate a lower bound on the optimal project completion time. We eliminate the node if the lower bound is no smaller than the best known upper bound.

If all nodes are eliminated at any level, then we backtrack to the previous level. We terminate whenever we reach the root node.

We now explain our lower bounds. Our lower bounds are found by using the Carlier and Rinnooy Kan's (C&R) Algorithm. For the node with set of assigned tasks, S and unassigned tasks, $\bar{S}$, we let

$$c_i = \sum_i c_{i,j} x_{i,j} \qquad \text{if} \quad i \in S$$

$$p_i = \sum_i p_{i,j} x_{i,j} \qquad \text{if} \quad i \in S$$

$$c_i = c_{i,m_i} \qquad \text{if} \quad i \in \bar{S}$$

$$p_i = p_{i,1} \qquad \text{if} \quad i \in \bar{S}$$

where first and m_ith (last) modes of activity i are the minimum processing time and minimum resource consumption modes, respectively.

Note that p_i and c_i, for $i \in \bar{S}$, is a lower bound on the respective processing time and resource requirement of task i. Hence an optimal project completion time found using lower bounds on the processing time and resource requirements, provides a lower bound on the actual optimal project completion time. The optimal project completion time over any defined set of task modes is found by using the C&R Algorithm.

Example 4.1: Consider an example instance whose precedence relations are given in Table 4.1 and illustrated by AoN diagram as in Figure 4.2.

Table 4.1 The Example Project Data

Activity	Immediate Predecessors	Number of modes
1	-	1
2	1	2
3	1	3
4	2	4
5	2	1
6	4, 5	2
7	3,5	4
8	6, 7	1

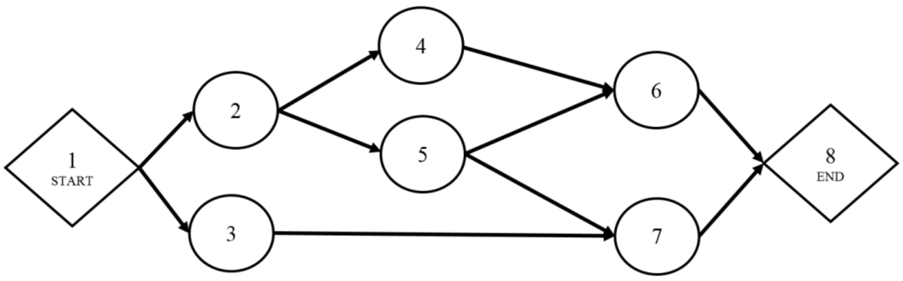

Figure 4.2 AoN Representation of Example Project

According to the number of modes for each task, processing times and resource consumptions are generated and given below, in Tables 4.2 and 4.3, respectively:

Table 4.2 The Processing Times of the Activities

Tasks	Mode 1	Mode 2	Mode 3	Mode 4
1	0	0	0	0
2	4	5	0	0
3	6	9	10	0
4	3	5	7	9
5	2	0	0	0
6	1	6	0	0
7	2	3	4	10
8	0	0	0	0

Table 4.3 The Resource Consumptions of the Activities

Tasks	Mode 1	Mode 2	Mode 3	Mode 4
1	0	0	0	0
2	3	2	0	0
3	8	5	2	0
4	10	9	5	4
5	5	0	0	0
6	10	2	0	0
7	7	5	3	1
8	0	0	0	0

There are two resource releases at time 2 and time 10 with the amounts of 10 and 15 units, respectively, i.e., B_2=10 and B_{10}=15.

Assume a partial solution depicted in Figure 4.3 where first mode is assigned to Task 2 (and first mode is assigned to Task 1). We call this partial solution as *node 1.1*. The cumulative resource supply line of *node 1.1* is B_t. The original and shifted cumulative resource requirement profiles are denoted as dotted and straight R_t lines, respectively, as in Figure 4.3. We will refer to Figures 4.3 and 4.4, for some future illustrations.

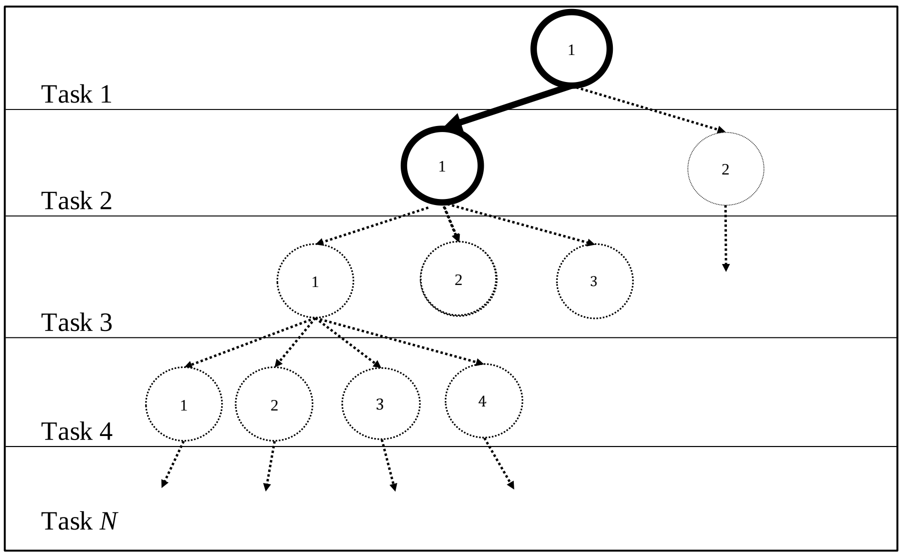

Figure 4.3 The BAB Tree Illustration of *Node 1.1*

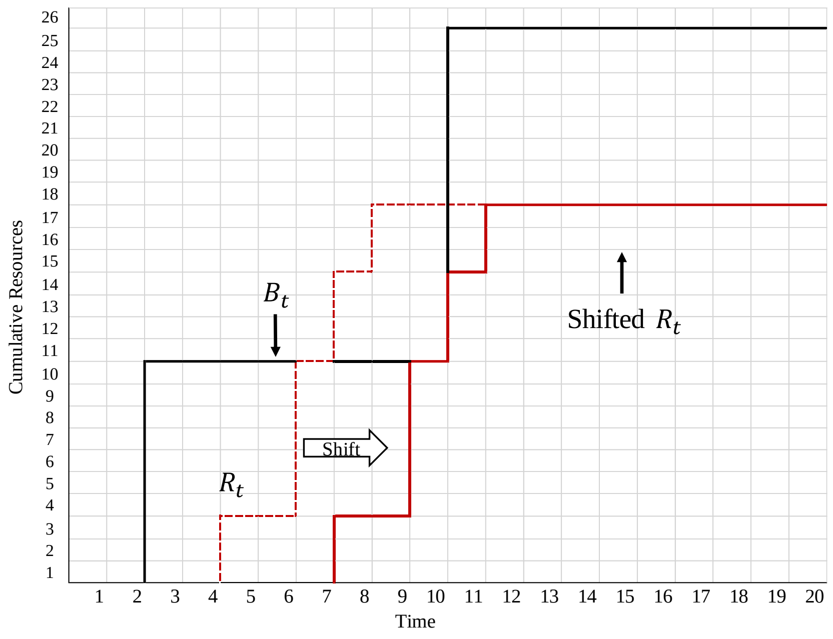

Figure 4.4 The Resource Profiles at *Node 1.1*

After finding the lower bound of *node 1.1*, we then proceed to the next level for the mode assignment of Task 2. *Node 1.1.2* represents the assignment of second mode to Task 2.

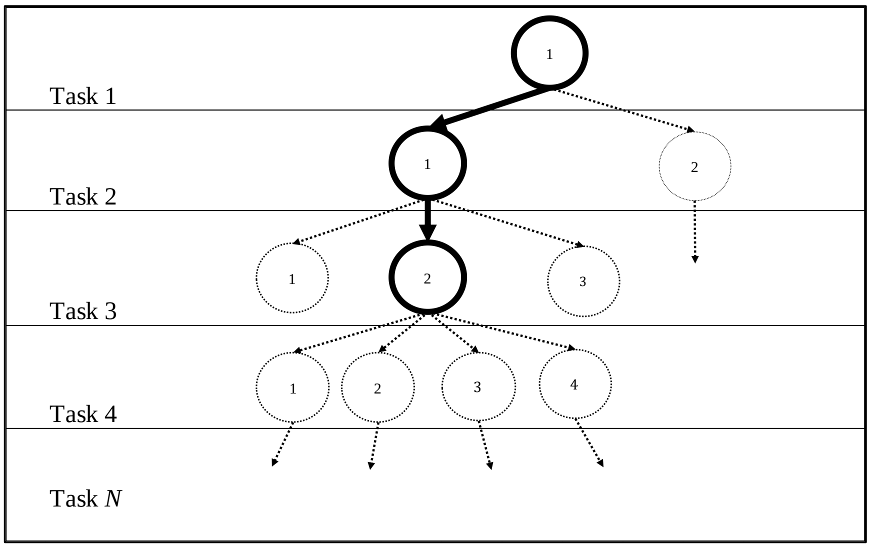

Figure 4.5 The BAB Tree Illustration of *Node 1.1.2*

Bold lines in Figure 4.5 shows the partial schedule where modes 1, 1 and 2 are assigned to tasks 1, 2 and 3, respectively. We use the actual processing times and resource consumptions of Tasks 1, 2 and 3 according to their respective mode assignments. For all other not-yet-assigned tasks, we use the minimum resource consumptions and processing times. So, our input to the C&R algorithm is as in Table 4.4.

Table 4.4 Input for the C&R Algorithm

Activitiy	1	2	3	4*	5*	6*	7*	8*
Mode	1	1	2	1	1	1	1	1
Process time	0	4	9	3	2	1	2	0
Resource consumption	0	3	5	4	5	2	1	0

*Not-yet assigned tasks

Using these data, the C&R algorithm finds the cumulative resource supply profile and the cumulative resource requirement profile of given activity modes. The profiles are shown in Figure 4.6. Recall that, the cumulative resource requirement line should be on or below the cumulative resource supply line. If not, it should be shifted to the right till it is on or below the cumulative resource supply line. Note that, in our example, R_t is originally not below B_t and should be shifted to the right by one unit to satisfy this. The completion time of this shifted optimal schedule, hence a lower bound of our current node, *node 1.1.2*, is 12 time units.

Note that, for *node 1.1*, p_3 and c_3 are at their minimum possible values, 6 and 2, respectively. When we proceed *node 1.1.2*, both p_3 and c_3 take their actual respective values of 9 and 5. These changes in timing and amount of cumulative resource requirement makes R_t of *node 1.1* in Figure 4.4 significantly different than R_t of *node 1.1.2* in Figure 4.6. We next discuss that, if one of p_3 or c_3 stays the same, the C&R algorithm may not be applied explicitly to arrive at a lower bound.

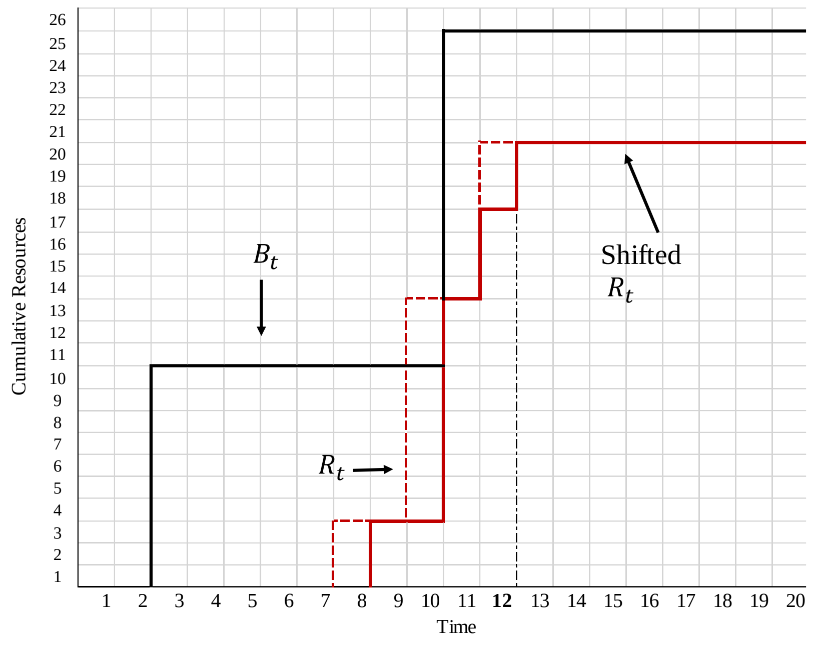

Figure 4.6 Illustration of the C&R Algorithm for Example 4.1

Note that at any node, a lower bound is found by setting all tasks to their smallest processing time and lowest resource consumption modes. Having known this solution, a lower bound on the cost of assigning task *i* to its shortest time, hence highest resource consumption mode, does not require the use of the C&R algorithm. Only simple shifting may help to find a lower bound, i.e., the resource requirement profile is updated by increasing the resource consumption of task *i* by $(c_{i,1} - c_{i,m_i})$ resource units.

In Figure 4.7, consider the node where we assign the highest resource consumption, -thereby smallest processing time- mode, i.e., the first mode, to Task 3. We now find the lower bound of that node without applying the C&R algorithm.

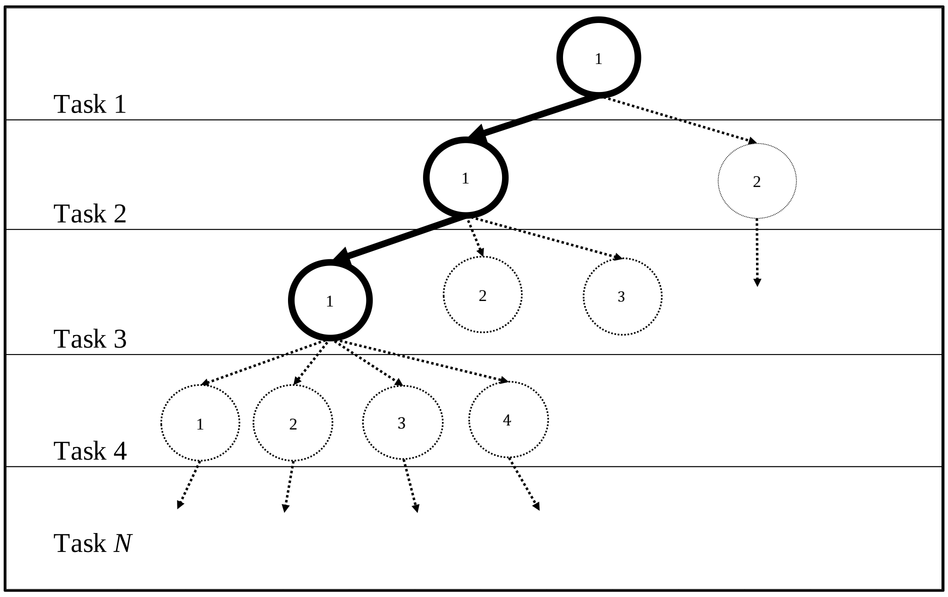

Figure 4.7 BAB Tree Illustration of the Next Node of Example 4.1

Cumulative resource requirement profile of the first two tasks with assigned modes {1, 1} and the remaining tasks with dummy modes is constructed and illustrated by R_t in Figure 4.4. So, $p_3 = 2$ and $c_3 = 2$ are used for the Task 3 while constructing the resource requirement profile. Now we are at the 1st mode node of Task 3 and change the resource consumption and process times with the actual parameters as $p_3 = 2$ and $c_3 = 8$. Due to the fact that processing time stays the same, the timings in the resource requirement profile does not change. The only change occurs at the resource requirement of Task 3 that affects the cumulative resource requirement of the tasks 3 through *N*. Thus, the resource requirement increases by 6 units, since $c_{3,1} - c_{3,m_3}$ is 6 as of time 9. We know the timings from the existing profile of *node 1.1* and this profile is by 6 units from time 9 as illustrated with the dotted line in Figure 4.8. Since the shifted line is now partially above the cumulative resource supply line, it has to be shifted to the right until it is fully on or below the B_t line. One may see from Figure 4.8 that, one unit of shift is sufficient to ensure feasibility, hence the optimality. Solid R_t line shows the shifted line. Hence, we find the lower bound as 12 by simply shifting the existing resource profile.

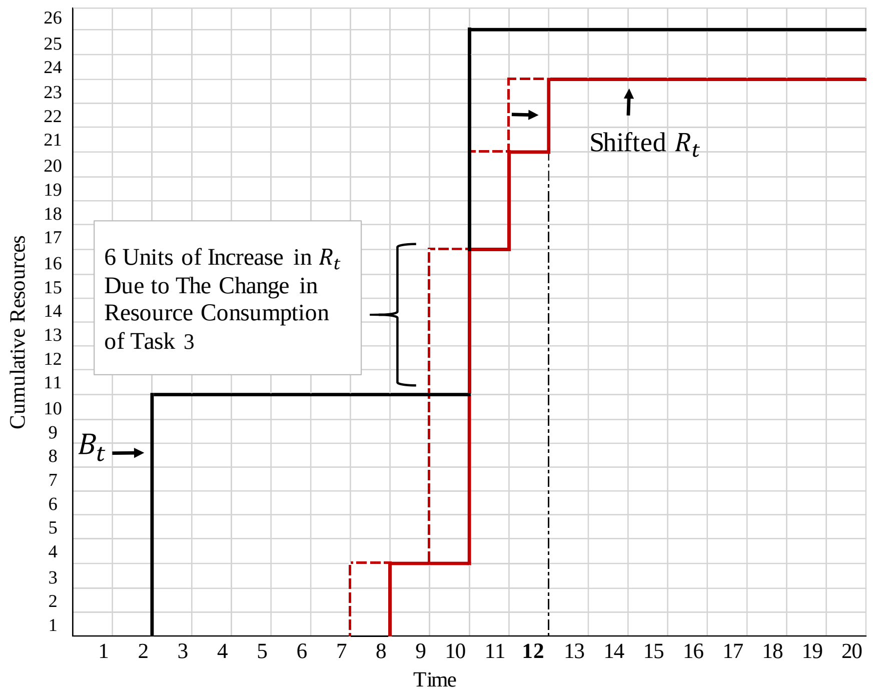

Figure 4.8 Illustration of Lower Bound Calculation of the New Node

Moreover, finding a feasible solution where each task is assigned to its highest resource consumption,-thereby smallest processing time- mode requires only updating the resource requirement profile by ($c_{max,\, i} - c_{min,\, i}$) units, for each task i. As we find the lower bound of the BAB node with Task 3 and mode1, we can obtain the complete solution with minimum processing time modes by shifting an existing cumulative resource requirement profile.

The following expression gives a lower bound on the optimal project completion time when task i is assigned to mode j:

$LB_{i,j}$ = Max { 0, $p_{i,j} - p_{i,1} - (TS_i + A)$ } + LF+A where

TS_i = Total slack time when task i is assigned to its minimum processing time mode

A = Total amount of shift over the earliest completion time of the late start schedule due to resource usages

LF = The completion time of the late start schedule of the previous node which uses $p_{i,\,1}$

Note that, (TS_i + A) is the amount that the processing time of task i can be increased without increasing the total project completion time.

The actual amount of increase over the minimum processing time is ($p_{i,\,j} - p_{i,\,1}$) time units. If ($p_{i,\,j} - p_{i,\,1}$) is greater than ($TS_i + A$), there will be ($p_{i,\,j} - p_{i,\,1}$) – (TS_i + A) units of increase in the completion time of the late start schedule. Otherwise, the increase in the processing time value does not affect the overall project completion time. To illustrate, we use *node 1.1.2* as in Figure 4.6. On the figure, we show the lower bound computation using only our total slack equation.

Recall that, we need the latest finish time and the total slack time of Task 3 that are calculated for *node 1.1*. We also make use of the cumulative resource requirement profile of *node 1.1*.

For *node 1.1,* we calculate the total slack times for all activities as follows:

$Total\ slack_i = LF_i - ES_i - p_i$

$TS_3 = LF_3 - ES_3 - p_3 = 6 - 0 - 6 = 0$

We now illustrate the computation of A.

Let A_1 be the amount of required shift of the cumulative resource requirement profile of *node 1.1*. When the resource requirement of Task 3 is updated according to its respective value in new level, the cumulative resource requirement profile of *node 1.1* is also updated. A is the amount of required shifts of the updated cumulative resource requirement profile of *node 1.1. A* is either equal to A_1, if A_1 is sufficient to maintain the resource profile feasibility for the updated resource profile, or greater than A_1, if some extra shift is needed.

For our example instance, A_1 is 3 units (see Figure 4.4). To find A, we update the cumulative resource requirement profile of *node 1.1*, by using the resource requirement of Mode 2 of Task 3, in place of its minimum resource requirement. The updated cumulative resource requirement profile is shown by the dotted line R_t, in Figure 4.9 below:

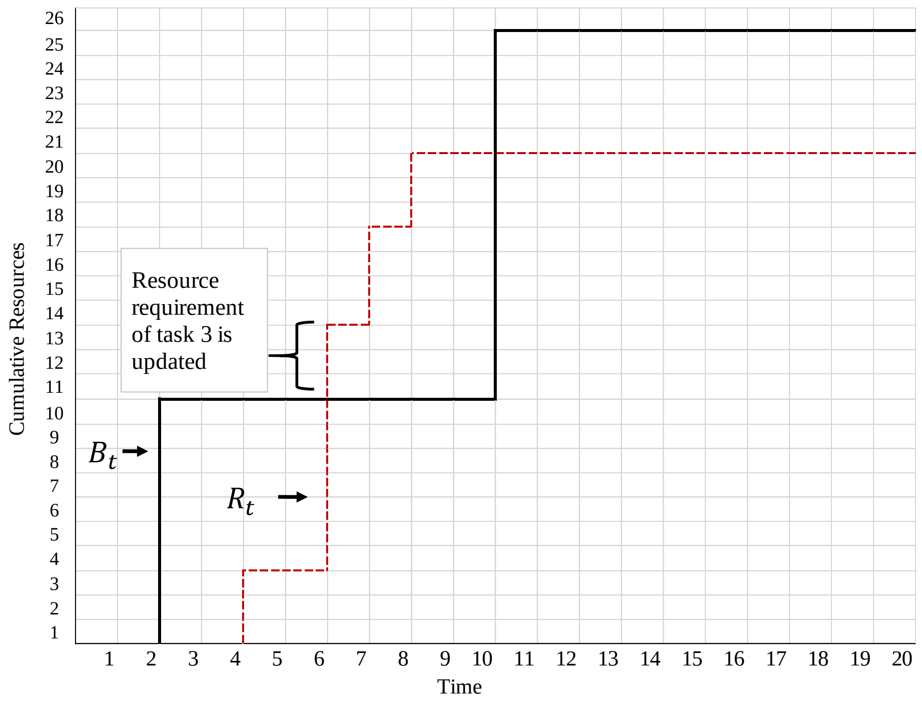

Figure 4.9 The Updated Resource Profile of *Node 1.1* Before the Shift

As Figure 4.9 implies, four units of shift is necessary, i.e., $A = 4$. The R_t is shifted by 4 units as shown in Figure 4.10. Now, all the components of our lower bound expression are available;

$$LB_{3,2} = \text{Max}\ \{\ 0,\ \ p_{3,2} - p_{3,1} - (TS_3 + A)\ \} + LF + A$$

$$LB_{3,2} = \text{Max}\ \{\ 0,\ \ 9 - 6 - (0 + 4)\ \} + 8 + 4 = 12$$

Note that, we find the same solution of Example 4.1 without referring to the C&R algorithm. However, we cannot guarantee on exact solution since we only consider the total slack time of the current activity. In latter levels of the BAB tree, total slack times of the predecessors of the current activity will be also determinant on total amount of shift. That is why, what we suggest is as a lower bound on the exact solution of the partial schedule, which may give the exact solution depending on a particular instance. Our example instance is one of those instances where the resulting lower bound coincides to an optimal solution.

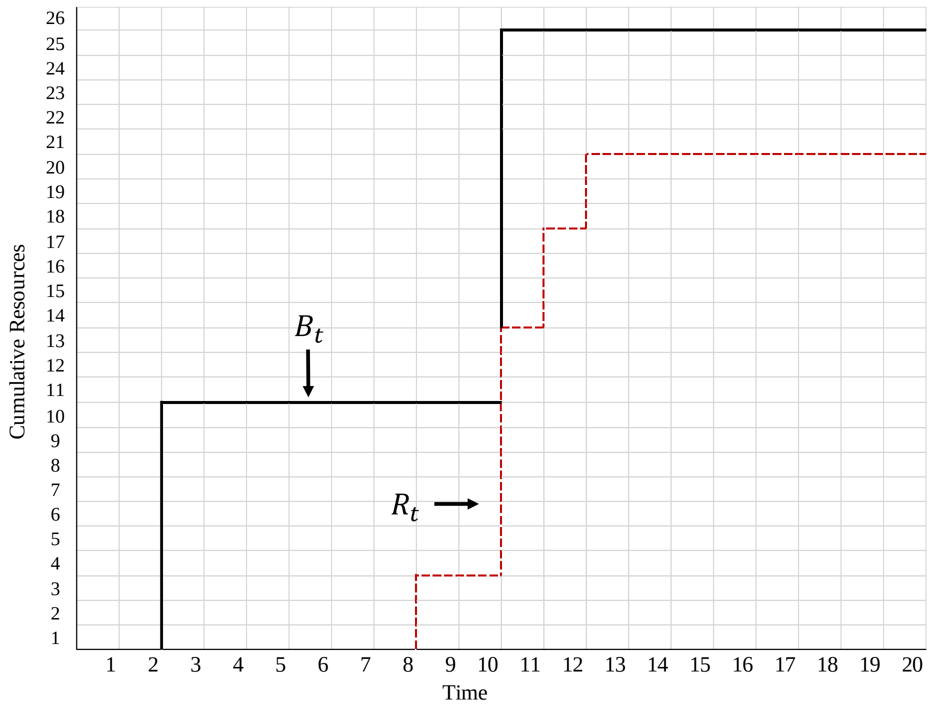

Figure 4.10 The Updated Resource Profile of *Node 1.1* After the Shift

At level *i*, for task *i*, while assigning to mode *j*, the following path is followed:

LB_i = lower bound found for the tasks from 1, 2,…,*i*-1 with fixed modes and the others at $(p_{i,1}, c_{i,m_i})$

$LB^1_{i,j}$ = lower bound found by C&R algorithm

$LB^2_{i,j}$ = Max { 0, $p_{i,j} - p_{i,1} - (TS_i + A)$ } + LF+A

If j = 1, then apply simple shifting to find $LB^1_{i,1}$

If $LB^1_{i,j} \geq UB$, then fathom the node that represents assigning mode 1 to task i

If $j \geq 2$, then find $LB^2_{i,j}$

If $LB^2_{i,j} \geq UB$, then fathom the node else find $LB^1_{i,j}$

If $LB^1_{i,j} \geq UB$, then fathom the node

At some modes, we update *UB* by simply assigning all tasks to their shortest time modes, i.e., by shifting the resource profile to the right. Let UB_2 be the resulting project completion time value.

If $UB_2 = LB$, then fathom the node by assigning tasks i+1 through N to their first modes.

If $UB_2 < UB$, then update *UB* and continue without fathoming.

CHAPTER 5

COMPUTATIONAL EXPERIMENTS

We design an experiment to test the performance of our solution approaches, i.e., mathematical model and branch and bound algorithm along with the bounding and mode elimination procedures.

In this chapter, we first give our data generation scheme and then discuss the results of our computational study. While analyzing the results, we first state the performance measures and then present the results of our preliminary and main experiments.

5.1 Data Generation

The structures of the project networks directly affect the difficulty of the solution. For example, a project with a serial flow, i.e., each activity has a single predecessor and single successor, (see Figure 5.1), is solved, trivially. The network complexity is an important factor of many computational experiments.

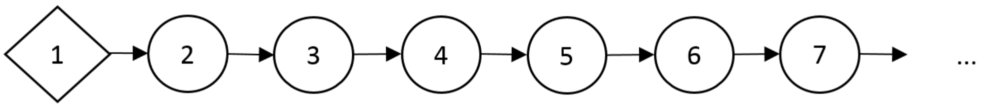

Figure 5.1 AoN Representation of the Example Serial Flow Project

Two measures are commonly used to define the complexity of the project networks. The first one is the coefficient of network complexity (CNC), introduced by Pascoe (1966) and simply defined as the number of arcs divided by the number of nodes. Thus, the higher the CNC, the more connected the network. Hence, for a fixed number of activities, an increase in CNC increases the complexity of the solutions.

Second commonly used network complexity measure is the complexity index (CI). The need for another measure arises from the possibility to construct networks of equal number of arcs and nodes but varying degrees of difficulty. So, along with CNC, we use CI in order to better discriminate project network complexity. CI is introduced by Bein et al.(1992) and is defined as the number of node duplications needed to transform an Activity on Arc (AoA) network into a series or parallel network. It basically measures the closeness of a network to a series-parallel one. The related studies so far have shown that an increase in CI, increases the network complexity and the solution time.

We take our precedence networks from Akkan et al. (2005). These networks have already specified CNC and CI values. We select 9 networks from Akkan et al.'s (2005) data set. The parameters of the selected networks are as stated below:

Table 5.1 The Network Parameters

Network Name	*N*	CI	CNC
N_1	10	0	2
N_2	20	0	2
N_3	30	0	2
N_4	35	5	2
N_5	40	13	2
N_6	50	13	5
N_7	60	13	5
N_8	85	13	5
N_9	100	13	6

We generate the other parameters as explained below:

Number of modes: The number of modes (m_i) for each activity (i) is generated from a discrete uniform distribution between 1 and 4. We set the lower limit to 1, as some activities may have only one option for processing. We set the upper limit to 4, as usually there may not be too many options of realizing an activity. Consider an excavation activity for a construction project. The alternative ways for performing this activity is usually limited to outsourcing from two different agencies or putting extra capital or using own resources.

Mode parameters: Resource consumption and activity processing times are generated from a discrete uniform distribution between 1 and 20 with no repetition. Activity durations and resource consumptions are sorted in an ascending order and a descending order, respectively, so that the modes are non-dominated.

Upper Bound on the Optimal Project Completion Time, T_{UB}: Note that setting an appropriate upper bound on the optimal project completion time plays an important role on the complexity of the mathematical model. The constraint sets (4) through (6) and the decision variables x_{it} are directly affected by the magnitude of T_{UB}. If T_{UB} is set too big, then there will be unnecessarily many constraints. If it is set too small, there may be infeasibilities. By recognizing this trade-off, we make use of the upper bound proposed in Chapter 3.6.1. Recall that to find T_{UB}, we evaluate the following four feasible schedules: minimum activity time, minimum resource usage and median activity time schedules each with objective function values Z_1, Z_2, Z_3, Z_4. Thus;

T_{UB} = Minimum $\{Z_1, Z_2, Z_3, Z_4\}$

For each instance, we find T_{UB} and set T to T_{UB}. Hence, the deadline differs for each instance depending on the minimum feasible schedule found.

For BAB algorithm, we do not want T as a constraint, so we use nonbinding T value. We first sum the maximum processing times of all tasks for all 10 instances. Maximum of these values is summed with the latest resource release time. So, we ensure that T is big enough that it is not a constraint on the solutions.

Resource releases: Resource release has two components: resource release amount and resource release time.

Resource release amount: Recall that, in the absence of resource constraints, the problem can be solved by the CPM. Hence, the amount of available resources directly affects the complexity of solutions. To observe this effect, we construct three different resource profiles as large, average and small. We know that the first mode of an activity i, has the minimum processing time $p_{i,1}$ and maximum resource consumption, $c_{i,1}$. Correspondingly, m_i, the last mode generated, has the maximum processing time $p_{i,\, m(i)}$ and minimum resource consumption $c_{i,\, m(i)}$. Let b_{max} and b_{min} be the total resource requirements if all activities are executed in their maximum and minimum resource consumption modes, respectively.

$b_{max} = \sum_{i=1}^{N} c_{i,1}$

$b_{min} = \sum_{i=1}^{N} c_{i,mi}$

Using b_{max} and b_{min} values, we find the total resource release amount for each profile as below:

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

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

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

Resource release times: When we consider our nonrenewable resource as money, we can expect that resource requirement of initial phases is greater than the final phases as many real life projects require the capital investment that of to initiate. To quantify these initial phases, we use a lower bound found by the Critical Path Method (CPM). Under unlimited resource assumption, CPM finds the completion time of a project with single mode activities. So, we solve the CPM for the first activity modes (with minimum processing time and zero resource consumption). The resulting project completion time is an obvious lower bound for the optimal project completion time, $CPMLB_{pmin}$. Assume 5 release times, $t_{1,2,3,4,5}$ (t_1 is the beginning of the project) that resource releases are to be scheduled. When the resulting time points are not integers, we simply round down as stated below:

$t_1=0$

$t_i = t_{i-1} + \lfloor \frac{CPMLBpmin}{5} \rfloor$ for $i = 2, 3, 4, 5$

For instance, when $b_{average}$ is 880 and $CPMLB_{pmin}$ is 74, the resources are released as in Figure 5.2 below:

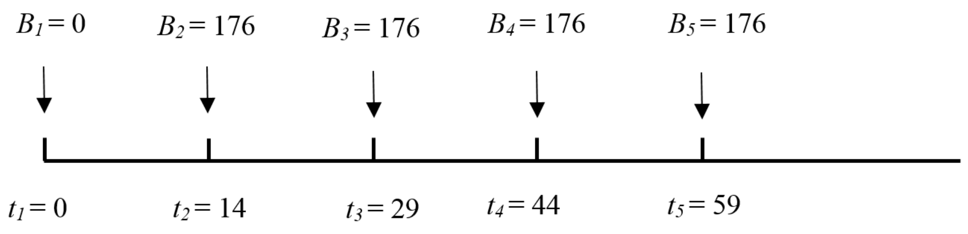

Figure 5.2 Illustration of Resource Releases for the Example Instance

We set a termination time limit of 2 hours, since our problem does not require immediate solutions as in many operational level problems like machine scheduling.

We solve the mathematical models with IBM ILOG CPLEX 12.6. The BAB algorithm is coded in C++ using Microsoft Visual Studio 2012. All the experiments are conducted on a computer with Intel(R) Core(TM) i7-4790 CPU @ 3.60 GHz, 8 GB RAM and Windows 10.

5.2 Performance Measures

In this subchapter, we present our performance measures that we use to evaluate the performance of our mathematical model and BAB algorithm, along with their modifications.

For the mathematical model, we use three performance measures:

1) Average Central Processing Unit (CPU) Time (in seconds)

2) Maximum Central Processing Unit (CPU) Time (in seconds)

3) Number of Unsolved Instances, out of 10, in our termination limit of 2 hours.

We use five performance measures for the BAB algorithm:

1) Average Central Processing Unit (CPU) Time (in seconds)

2) Maximum Central Processing Unit (CPU) Time (in seconds)

3) Average Number of Nodes

4) Maximum Number of Nodes

5) Number of Unsolved Instances, out of 10, in our termination limit of 2 hours.

5.3 Preliminary Experiments

We perform some preliminary tests to see the effect of some mechanisms on the performance of the solution approaches.

5.3.1 Mathematical Model

We aim to see the effects of the bounds on the activity completion times and mode elimination properties on the performance of the mathematical model. Basically, we want to see whether the extra effort spent to prepare and test the mechanisms is lower than the amount of reductions they would bring. In other words, we want to see whether it is worth to use these mechanisms.

For the preliminary experiments, we select N = 10 activities, and three different resource profiles: large, average and small. We generate 10 instances for each resource profile, hence solve a total of 30 instances.

We test the following four versions of the mathematical model.

M_1: original model (without bounds and elimination mechanisms)

M_2: model with bounds but without elimination mechanisms

M_3: model with elimination mechanisms and without bounds

M_4: model with bounds and elimination mechanisms

Table 5.2 reports on the solution times of the model versions for large, medium and small resource release amounts. The table includes the average and maximum solution times of 10 problem instances and the number of instances that remain unsolved after 2 hours.

Table 5.2 The CPU Times (in Seconds) of the Mathematical Model

N = 10

	Large		Medium		Small	
	Avg	*Max*	*Avg*	*Max*	*Avg*	*Max*
M_1	0.55	1.30	2.57	8.19	8.15	34.30
M_2	0.47	1.13	1.78	8.72	5.45	16.22
M_3	0.55	1.27	2.62	14.70	7.63	23.41
M_4	0.37	0.86	1.01	4.47	3.22	7.58

The table reveals that incorporating the bounds on the activity completion times significantly improves the performance of the model. Note that for large resource profile case, the average CPU time reduces to 0.47 from 0.55 seconds with the incorporation of the bounds. These are more significant for medium and small profiles where the CPU times are higher. For medium resource profiles, the average CPU time reduces to 1.78 from 2.57 seconds, whereas the reduction is to 5.45 from 8.15 seconds for small resource profile case. On the other hand, the mode elimination mechanisms are not that effective. When the resource profiles are large and medium, i.e., the problem instances are easier to solve, it is not worth to incorporate the mode elimination mechanisms. However, when the instances are harder to solve, i.e., the resource profiles are small, then there is a slight reduction in the CPU times (the average CPU time reduces to 7.63 from 8.15 seconds). We obtain the most significant reduction when the bounds and mode elimination mechanisms are used together. The CPU times reduce to 0.37, 1.01 and 3.22 seconds from 0.55, 2.57 and 8.15 seconds for large, medium and small resource profiles, respectively.

Using those results, we decide to perform our main experiment with larger-sized problem instances using the bounds and mode elimination mechanisms.

5.3.2 Branch and Bound Algorithm

We design eight versions of the branch and bound algorithm to see the effects of the lower bound on the complexity of the solutions.

We use the following notation to state the versions of the branch and bound algorithm.

S=Set of assigned tasks

$\bar{S}$=Set of unassigned tasks

The BAB versions are as stated below:

1. BAB_1 = BAB where the lower bounds are found as follows:

   $c_i = \sum_i c_{i,j} x_{i,j}$ if $i \in S$

   $p_i = \sum_i p_{i,j} x_{i,j}$ if $i \in S$

   $c_i = 0$ if $i \in \bar{S}$

   $p_i = 0$ if $i \in \bar{S}$

2. BAB_2 = BAB where the lower bounds are found by using Carlier and Rinnooy Kan's (C&R) algorithm. For the node with set of assigned tasks, *S* and unassigned tasks, $\bar{S}$, we let

   $c_i = \sum_i c_{i,j} x_{i,j}$ if $i \in S$

   $p_i = \sum_i p_{i,j} x_{i,j}$ if $i \in S$

   $c_i = c_{i,m_i}$ if $i \in \bar{S}$

   $p_i = p_{i,1}$ if $i \in \bar{S}$

   where first and m_ith modes of activity *i* are the minimum processing time and minimum resource consumption modes, respectively.

3. BAB_3 = BAB algorithm that uses the following lower bound for activity *i* and mode *j*:

   $LB_{i,j} = LF + A$

   where *LF* is the earliest completion time of the late start schedule and *A* is the total shift of the earliest completion time of the late start schedule.

4. BAB_4 = BAB that uses the following lower bound for activity *i* and mode *j*:

   $LB_{i,j}$ = Max { 0, $p_{i,j} - p_{i,1} - (TS_i + A)$ } + $LF+A$

The other four versions of the BAB algorithm, BAB_5 through BAB_8 look for the effect of the following local optimality check on the performance.

Recall that finding a partial schedule is possible where the task at the next level is assigned to its highest resource consumption and smallest processing time mode by further shifting the resource requirement profile by $(c_{max,\, i} - c_{min,\, i})$ units, for the new task *i*. Using the similar approach, we can find a feasible solution where each task is assigned to its highest resource consumption,-thereby smallest processing time- mode by only shifting the resource requirement profile by $(c_{max,\, i} - c_{min,\, i})$ units, for each task *i*. In the next four BAB versions, we check the feasibility at some particular levels by updating the resource profile. A feasible solution is the optimal solution for that particular node. For example, in Figure 5.3 mode assignments of tasks 1, 2 and 3 are done. Before proceeding to Task 4, all other tasks are assigned to their first modes. If the resulting assignment is feasible, then corresponding assignment is optimal for the current node since minimum processing times are used. Hence, the total number of nodes decreases. Note that, once we cannot find a feasible solution, the check is useless and kind of creating an additional computational burden.

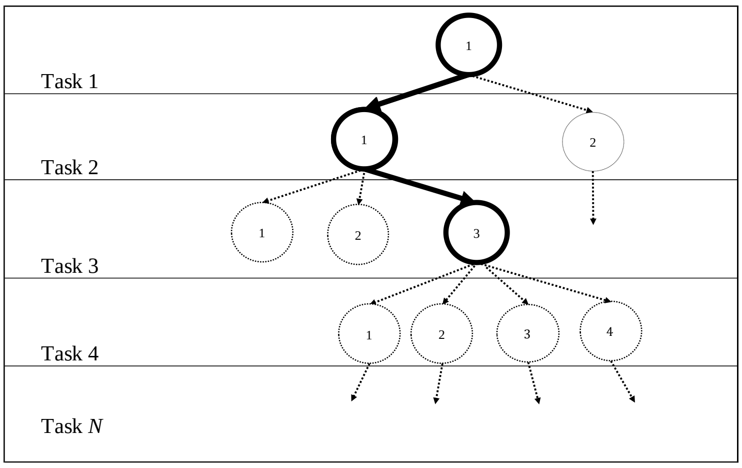

Figure 5.3 Illustration of an Example Node from BAB Tree

5. BAB_5 = BAB that checks the local optimality starting from the 90%th level of the BAB tree. For example, for N=30, it starts checking the local optimality from the 27th level and checks the levels 27, 28, 29.
6. BAB_6 = BAB that checks the local optimality only for the last one-third of the BAB tree. For example, for N=30, it starts checking the local optimality starting from the 20th level. So it starts checking earlier than the BAB_5.
7. BAB_7 = BAB that checks the local optimality starting from the first one-third level of the BAB tree.
8. BAB_8 = BAB that checks the local optimality for all levels.

Tables 5.3 and 5.4 give the CPU times and the average number of nodes of each version, respectively.

Table 5.3 The CPU Times of the BAB Versions

N = 30

	Large		Medium		Small	
	Avg	*Max*	*Avg*	*Max*	*Avg*	*Max*
BAB_1	1481.84	5559.92	1470.48	4104.94	890.02	2874.47
BAB_2	0.17	1.14	1.86	12.28	3.02	10.33
BAB_3	0.22	1.69	2.64	17.54	4.27	14.53
BAB_4	0.12	0.78	1.49	9.54	2.66	8.43
BAB_5	0.12	0.81	1.50	9.55	2.66	8.45
BAB_6	0.13	0.92	1.73	11.27	3.01	9.86
BAB_7	0.13	0.95	1.76	11.28	3.08	10.01
BAB_8	0.14	0.95	1.77	11.30	3.11	10.14

Table 5.4 The Average Number of Nodes of The BAB Versions

N = 30

	Large	Medium	Small
BAB_1	128832943	135005088	82509833
BAB_2	12833	163139	267593
BAB_3	12665	162898	266786
BAB_4	6387	85747	156414
BAB_5	6377	85734	156401
BAB_6	6378	85735	156401
BAB_7	6378	85735	156401
BAB_8	6378	85735	156401

When the resource consumptions and processing times are changed to the minimum ones from zero, the average CPU times drastically decrease from 1481.84 to 0.17, 1470.48 to 1.86, 890.02 to 3.02 seconds, for large, medium and small resource profiles, respectively. When the lower bound calculation with the earliest completion time of the late start schedule and the total shift of it is incorporated, as in BAB_3, the average CPU times increase to 0.17 from 0.22, 1.86

from 2.64, 3.02 from 4.27 seconds, even though the total number of nodes decrease from 12833.40

to 12665.30, 163139.70 to 162898.10, 267593.30 to 266786.50 for large, medium and small resource profiles, respectively. Note that, the additional effort spent to calculate the lower bound does not help to improve the performance of the algorithm.

When the local optimal checks are included in the last 90% of the levels of the BAB tree, the average CPU times stay as same at 0.12 and 2.66 seconds for the large and small resource profiles, respectively. On the other hand, the average number of nodes significantly decrease from 6387 to 6377 and 156414 to 156401 for the large and small resource profiles, respectively. For the medium resource profile, the average CPU time slightly increases from 1.49 to 1.50 seconds.

For the last three BAB versions, i.e., BAB_6, BAB_7 and BAB_8, the average number of nodes is the same with that of BAB_5. It implies that all the node eliminations due to attaining the local optimal are in the last 90% of the levels of the BAB tree. So the additional computation effort spent for more frequent checks, i.e., at each level in BAB_8, does not justify. Likewise, the average CPU times increase from 0.12 to 0.13, 1.50 to 1.73, 2.66 to 3.01as the check level changes from 90% to 66% for the large, medium and small resource profiles, respectively. The average CPU times become worse off, when the checks are done at each level, as the results of BAB_8 indicate.

Using those results, we decide to perform our main experiment with larger-sized problem instances using the BAB_5 version since BAB_5 is likely to lead to more drastic decreases in the number of nodes as the problem size increases. Considering the significant decrease in the average number nodes for all profiles, the slight increase in the average CPU time of the medium resource profile is negligible.

We also observe from the results that the resource profiles are significant in terms of the solution times. When the resource profiles are small, i.e., the resources are scarce, the resource distribution problem becomes harder to solve. On the other

hand, when the resource profiles are large, the performance of the lower bounds significantly increases.

5.4 Main Experiments

5.4.1 Mathematical Model

Through preliminary runs on small sized instances, we observe that using mode eliminations and activity completion time bounds significantly improves the performance of the model. Hence we perform our main runs on larger sized instances, using those mechanisms. For three different resource profiles and project networks and ten instances for each, we test the performance of our mathematical model on 90 problem instances. We report the results for N=10, 20 and 30 in Tables 5.5 and 5.6. Table 5.5 gives the number of instances that could not solved in our termination limit of 2 hours. We perform our main experiments for the mathematical model using M_4, the model with bounds and mode elimination mechanisms.

Table 5.5 Number of Unsolved Instances out of 10 in 2 Hours

N	Large	Medium	Small
10	0	0	0
20	0	2	5
30	9	10	10

To see the effect of the termination time on the number of solved instances, we increase the time limit to 20 hours. After 20 hours, we end up with a single unsolved instance for medium resource profile and three unsolved instances for small resource profile. Hence, increasing the limit above 2 hours would not be much effective for the guarantee of optimality.

Table 5.6 gives the average and maximum CPU times, only for the instances that can be solved in 2 hours.

Table 5.6 The Average and Maximum CPU Times (in Seconds) of the Solved Instances

N	Large		Medium		Small	
	Avg	Max	Avg	Max	Avg	Max
10	0.37	0.86	1.01	4.47	3.22	7.58
20	5.62	12.56	14.48	37.38	334.57	1430.14
30	17.18	17.18	-	-	-	-

Note from Tables 5.5 and 5.6 that the performance of the model significantly deteriorates with the increase in the problem size. This is due to the increasing number of decision variables for mode assignments and completion times.

When *N* is increased from 10 to 20, the average CPU times increase from 0.37 to 5.62, 1.01 to 14.48 and 3.22 to 334.57 seconds, for large, medium and small resource profiles, respectively. When *N* becomes 30, the increases are more drastic, and majority of instances cannot be solved in 2 hours. Those results altogether verify the exponential nature of the model.

We also observe from the results that the resource profiles play significant role in the difficulty of the attaining optimal solutions. The reductions in the amount of resource releases adds to the complexity of the model. Note that for *N*=20, when the resource profiles are large, medium and small, the respective average CPU times are 5.62, 14.48 and 334.57 seconds.

When the resource profiles are small, i.e., the resources are scarce, the resource distribution problem becomes hard to solve one. On the other hand, when the resource profiles are large, the decisions related with resource distribution are taken much easily as the problem approaches to unlimited resource case.

5.4.2 Branch and Bound Algorithm

We now discuss the performance of our branch and bound algorithm. We test the performance of our algorithm on 450 problem instances. We report the results in Tables 5.7, 5.8 and 5.9 for the large, medium and small resource profiles, respectively. The numbers within the parentheses denote the number of solved instances over 10 instances.

Table 5.7 The Average and Maximum CPU Times (in Seconds) of the BAB Solutions with the Large Resource Profile

Network Size	CPU Times		Nodes	
	Average	Max	Average	Max
30	0.13	0.93	6378	46061
35	1.10	5.06	35497	188627
40	18.10	95.42	426460	2553897
50	28.78	128.44	667720	3062342
60 (8)	2728.02	7200.00*	24494879	103423573
85 (7)	3358.59	7200.00*	14460183	58345064
100 (6)	3781.82	7200.00*	12912836	31257975

* The termination limit of 2 hours is used for the unsolved instances

Table 5.8 The Average and Maximum CPU Times (in Seconds) of the BAB Solutions with the Medium Resource Profile

Network Size	CPU Times		Nodes	
	Average	Max	Average	Max
30	1.71	11.05	85735.50	558956.00
35	36.74	174.91	1495045.10	7986640.00
40	71.10	468.66	2575641.70	16957072.00
50 (7)	3704.54	7200.00*	47424312.29	113210258.00
60 (3)	5501.20	7200.00*	52176245.67	114562411.00

* The termination limit of 2 hours is used for the unsolved instances

Table 5.9 The Average and Maximum CPU Times (in Seconds) of the BAB Solutions with the Small Resource Profile

Network Size	CPU Times		Nodes	
	Average	Max	Average	Max
30	3.06	9.85	156411.50	508678.00
35	74.16	352.45	3049657.80	13642512.00
40	99.37	610.20	4022007.30	24643248.00

For all resource profiles, the increases in the number of tasks significantly affects the performance of the algorithm. When *N*=30, all instances could be solved in about 3 seconds. When *N*=50, some instances for medium profile and almost all instances for small profile could not be solved in 2 hours.

Another significant factor that affects the performance is the resource profile. When the resource profile is large, the instances are much easier to solve. This is due to the fact that the resource requirements are not playing a major role in mode selection and usually the modes having higher requirements are selected easily. That is why, mode selection process is not as involved. When the resource profile is small, then there is a big trade-off between time and resource requirements of the activities. Therefore, the mode selection process that affects the complexity is hard.

Note from the tables that for a moderate size problem with 35 tasks, the average CPU times are 1.1, 36.74 and 74.16 seconds, for large, medium and small resource profiles. When there are 50 tasks, all instances could be solved in almost 2 minutes (the maximum CPU time is 128.44 seconds) for large resource profile. For the medium resource profile, we observe CPU times approaching to 2 hours (the maximum CPU time for the solves instances is 6210.80 seconds) for 7 instances and 3 instances remain unsolved in 2 hours.

For large resource profiles, an optimal solution can be obtained to majority of the instances when the number of tasks is 100 or below. This limit is 50 tasks when the resource profile is medium and 40 tasks when the resource profile is small.

When the resource profiles are medium or small, for large sized problem instances, our suggestion would be to decompose the project network into subnetworks and solve each network to optimality by the branch and bound algorithm.

CHAPTER 6

CONCLUSIONS

In this thesis, we consider a project scheduling problem with a single nonrenewable resource. We assume that the resource is released in scheduled times at specified quantities and the resource is consumed at activity completions. The activities can be processed at different modes where a mode is defined by a processing time and a resource requirement amount. Our problem is to select the modes and timings of the activities so as to minimize the project completion time.

We formulate the problem as a mixed integer linear programming model. To reduce the number of integer variables we introduce bounds on the optimal values of the activity completion times and develop some mode elimination mechanisms. The results of our experiments have revealed that the complexity of the model solutions highly depends on the number of integer variables and the reduction mechanisms are very effective in reducing this complexity. Moreover, resource profiles play a significant role in the complexity of the solutions. When the resources are scarce, the resource distribution problem becomes hard to solve one. On the other hand, when the resource profiles are large, the decisions related with resource distribution are taken much easily as the problem approaches to unlimited resource case. We report that the complexity of the solution significantly increases, as the available amount of resource decreases. The project networks with 30 activities cannot be solved in our time limits even with the large resource profile which points out the need for more efficient optimization algorithm. Recognizing this fact, we present our branch and bound algorithm (BAB).

The results of our preliminary experiments show that the lower bounding mechanisms have significant effect on the performance of our BAB algorithm. The results of our main experiments reveal that the increases in the number of tasks significantly affects the performance of the algorithm for all resource profiles. Another significant factor that affects the performance is the resource profile. For large resource profiles, an optimal solution can be obtained in two hours for the majority of the instances when the number of tasks is 100 or below. This limit is 50 and 40 tasks when the resource profile is medium and small, respectively. Thus, we conclude that the scarcity of resource significantly increases the solution complexity.

In this thesis, we present exact solution methods to the problem. In the future, heuristic methods that find good solutions in reasonable times can be studied. The heuristic procedures may decompose the problem into subproblems where each subproblem is solved by our branch and bound algorithm. Moreover, beam search algorithms that take their spirit from our branch and bound algorithm might be worth-developing.

The future research may consider the generalization of our mixed integer linear programming model to all resource constrained programs including the renewable resources. The idea of connecting the mode decision and completion time variables using a continuous decision variable may be extended to all objective functions.

Another noteworthy research area might be to analyze the stochastic nature of the model with random resource release profile. Generating robust schedules that minimize the negative effects of uncertain events like late resource arrivals and few resource quantities may be worth-studying.

In our study, we ignore the time value of the nonrenewable resource. Incorporating the time value, hence dealing with discounted cash flows, might be an interesting future research direction.

REFERENCES

Akkan, C., Drexl, A. and Kimms, A. (2005). Network decomposition-based benchmark results for the discrete time-cost tradeoff problem. *European Journal of Operational Research, 165*(2), 339-358.

Alcaraz, J., Maroto, C., and Ruiz, R., (2003). Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms. *Journal of the Operational Research Society*, *54*(6), 614-626.

Azizoglu, M., Çetinkaya, F.C. and Pamir, S.K. (2015). LP relaxation-based solution algorithms for the multi-mode project scheduling with a non-renewable resource. *European Journal of Industrial Engineering, 9*(4), 450-469.

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

Blazewicz, J., Lenstra, J.K. and Rinnooy Kan, A.H.G. (1983). Scheduling subject to resource constraints: Classification and complexity. Discrete Applied Mathematics, 5(1), 11-24.

Brucker, P., Drexl, A., Mohring, R., Neumann, K. and Pesch, E., (1999). Resource constrained project scheduling: Notation, classification, models and methods. European Journal of Operational Research, 112(1), 3-41.

Brucker, P., Knust, S., Schoo, A. and Thiele, O. (1998). A branch and bound algorithm for the resource-constrained project scheduling problem. European Journal of Operational Research, 107, 272-278.

Carlier, J. and Rinnooy Kan, A.H.G. (1982). Scheduling subject to nonrenewable-resource constraints. *Operations Research Letters, 1*(2), 52-55.

Chaleshtarti, A.S. and Shadrokh, S. (2014). A branch and cut algorithm for resource-constrained project scheduling problem subject to nonrenewable resources with pre-scheduled procurement. *Arabian Journal for Science and Engineering*, *39*(11), 8359-8369.

De, P., Dunne, E.J., Gosh, J.B. and Wells, C.E., (1995). The discrete time-cost trade-off problem revisited. *European Journal of Operational Research, 81*(2), 225-238.

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

Değirmenci, G., and Azizoğlu, M. (2013). Branch and bound based solution algorithms for the budget constrained discrete time/cost trade-off problem. *Journal of Operational Research Society, 64*(10), 1474-1484.

Demeulemeester, E., De Reyck, B., Foubert, B., Herroelen, W. and Vanhoucke, M. (1998). New computational results on the discrete time/cost trade-off problem in project networks. *The Journal of the Operational Research Society, 49*(11), 1153-1163.

Demeulemeester, E. and Herroelen, W. (1992). A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management Science, 38*(12).

Hafizoglu, A.B. and Azizoglu, M. (2010). Linear programming based approaches for the discrete time/cost trade-off problem in project networks. *The Journal of the Operational Research Society, 61*(4), 676-685.

Hartmann, S. and Briskorn, D. (2010). A survey of the variants and extensions of the resource constrained scheduling problem. *European Journal of Operational Research*, *207*(1),1-14.

Hartmann, S. and Drexl, A., (1998). Project scheduling with multiple modes: A comparison of exact algorithms. *Networks, 32*(4), 283-297.

Hazir, O., Haouari, M. and Erel, E. (2010). Discrete Time/Cost Trade-off Problem: A Decomposition Based Solution Algorithm for the Budget Version. *Computers and Operations Research, 37,* 649-655.

Herroelen, W.S., De Reyck, B. and Demeulemeester, E.L. (1998). Resource-constrained project scheduling: A survey of recent developments. *Computers and Operations Research, 25*(4), 279-302.

Herroelen, W. and Leus, R. (2005). Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research, 165,* 289-306.

Jozefowska, J., Mika, M., Rozycki, R., Waligora, G., ans Weglarz, J. (2001). Simulated annealing for multi-mode resource-constrained project scheduling. *Annals of Operations Research, 102*(1-4), 137-155.

Kelley, J.E., Jr. and Walker, M.R., (1959). Critical path planning and scheduling. *Proceedings of the Eastern Joint Computer Conference,* Boston, MA, 160-173.

Klastorin, T. (2004). *Project Management: Tools and Trade-Offs*. NJ: Wiley.

Kolisch, R. and Hartmann, S. (2006). Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research, 174*(1), 23-37.

Kolisch, R. and Padman, R. (2001). An integrated survey of deterministic project scheduling. *Omega: The International Journal of Management Science. 29*(3), 249-272.

Kolisch, R., Sprecher, A. and Drexl, A. (1995). Characterization and generation of a general class of resource-constrained project scheduling problems. *Management Science, 41*(11), 1693-1703.

Mingozzi, A., Maniezzo, V., Ricciardelli, S. and Bianco, L. (1998). An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. *Management Science, 44*(5), 714-729.

Ozdamar, L. and Ulusoy, G. (1994). A local constraint based analysis approach to project scheduling under general resource constraints. *European Journal of Operational Research*, *79*(2), 287-298.

Ozdamar, L. and Ulusoy, G. (1995). A survey on the resource-constrained project scheduling problem. *IEE Transactions, 27*(5), 574-586.

Pascoe, T.L. (1966). Allocation of resources - CPM. *Revue Fran~aise de Recherche Op~rationelle, 38,* 31-38.

Sabzehparvar, M. and Seyed-Hosseini, S.M. (2008). *The Journal of Supercomputing*, *44*(3), 257-273.

Skutella, M. (1998). Approximation algorithms for the discrete time-cost tradeoff problem. *Mathematics of Operations Research*, *23*(4), 909-929.

Slowinski, R. (1980). Two approaches to problems of resource allocation among project activities - a comparative study. *The Journal of the Operational Research Society*, *31*(8), 711-723.

Sprecher, A. and Drexl, A. (1998). Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm. *European Journal of Operational Research, 107*(2), 431-450.

Sprecher, A., Hartmann, S., and Drexl, A. (1997). An exact algorithm for project scheduling with multiple modes. *OR Spectrum, 19*(3), 195-203.

Talbot, F.B., (1982). Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case. *Management Science, 28*(10), 1197-1210.

Zhu, G., Bard, J.F. and Yu, G. (2006). A branch-and-cut procedure for the multimode resource-constrained project-scheduling problem. *INFORMS Journal on Computing*, *18*(3), 377-390.

Vanhoucke, M., (2005). New computational results for the discrete time/cost trade-off problem with time-switch constraints. *European Journal of Operational Research*, *165*(2), 359-374.

Van Peteghem, V. and Vanhoucke, M., (2009). An artificial immune system for the multi-mode resource-constrained project scheduling problem. *Lecture Notes in Computer Science*, *5482*, 85-96.

Vanhoucke, M., and Debels, D. (2007). The discrete time/cost trade-off problem: Extensions and heuristic procedures. *Journal of Scheduling*, *10*(4-5), 311-326.

Weglarz, J., Jozefowska, J., Mika, M. and Waligora, G. (2011). Project scheduling with finite or infinite number of activity processing modes - a survey. *European Journal of Operational Research*, *208*(3), 117-205.