RESOURCE INVESTMENT PROBLEM WITH TIME/RESOURCE TRADE-OFFS

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

BY

ERDEM ÇOLAK

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

JUNE 2011

Approval of the thesis:

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

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

Prof. Dr. Sinan Kayalıgil _____
Head of Department, **Industrial Engineering**

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

Examining Committee Members:

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

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

Assist. Prof. Dr. Z. Pelin Bayındır _____
Industrial Engineering Dept., METU

Assist. Prof. Dr. Sinan Gürel _____
Industrial Engineering Dept., METU

Assist. Prof. Dr. Banu Yüksel Özkaya _____
Industrial Engineering Dept., Hacettepe University

Date: 30.06.2011

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

Name, Last Name: ERDEM ÇOLAK

Signature :

ABSTRACT

RESOURCE INVESTMENT PROBLEM WITH TIME/RESOURCE TRADE-OFFS

Çolak, Erdem

M.Sc., Department of Industrial Engineering

Supervisor: Prof. Dr. Meral Azizoğlu

June 2011, 89 pages

In this study, we consider a resource investment problem with time/resource trade-offs in project environments. We assume each mode of an activity is characterized by its processing time and resource requirement and there is a single renewable resource. Our aim is to minimize the maximum resource usage, hence the total amount invested for the single resource.

We formulate the problem as a mixed integer linear model and find optimal solutions for small sized problem instances. We propose several lower bounding procedures to find high quality estimates on the optimal resource investment cost. We use our lower bounds to evaluate the performance of our heuristic procedures.

The results of our computational experiments have revealed the satisfactory performances of our lower bounds and heuristic procedures.

Keywords: Projects, Resource Investment Time/Resource Trade-off, Bounding Procedures

ÖZ

ZAMAN/KAYNAK ÖDÜNLEŞİMLİ KAYNAK YATIRIM PROBLEMİ

Çolak, Erdem

Yüksek Lisans, Endüstri Mühendisliği Bölümü

Tez Yöneticisi: Prof. Dr. Meral Azizoğlu

Haziran 2011, 89 sayfa

Bu çalışmada proje ortamlarında zaman/kaynak ödünleşimini dikkate alan bir kaynak yatırım planlaması problemini ele aldık. Aktivitelerin işlem zamanı ve kaynak gereksinimleri ile tanımlandıklarını ve tek bir yenilebilir kaynak olduğunu varsaydık. Amacımız en büyük kaynak kullanımını, dolayısıyla, toplam kaynak yatırımını enazlamaktır.

Problemi tam sayılı karmaşık model olarak formüle ettik ve küçük boylu projeler için optimal çözümleri bulabildik. Optimal kaynak maliyetlerine yakın sonuçlar üretebilmek amacıyla alt sınırlama yöntemleri geliştirdik. Alt sınırlarımızı sezgisel yöntemlerin verimliliğini ölçmek için kullandık.

Deneysel sonuçlarımız, alt sınırlar ve sezgisel yöntemlerimizin başarılı sonuçlar verdiğini göstermiştir.

Anahtar Kelimeler: Proje, Kaynak Yatırım, Zaman/Kaynak Ödünleşimi, Sınırlama Teknikleri

To My Mother, Father and Sister...

ACKNOWLEDGMENTS

First, and foremost, I would like to express my gratefulness to my supervisor Prof. Meral Azizoğlu for guiding me admirably, motivating me in hardest times and helping me in most tiring times. Without her great encouragement together with her amicable support, this thesis would not have been written.

Then, I should reveal that, being with me in all this exhaustive period and providing endless support, my family deserves the best wishes. Therefore, I would like to thank deeply to my mother Melek Çolak, my father Yılmaz Çolak and my sister Duygu for their never-ending understanding and love. Without their love and patience everything would be more difficult.

I would like to express my gratitude to my dear roommates in the past two years, Bilge Çelik and Volkan Gümüşkaya for firstly being great friends, and secondly for being great colleagues. Bilge's smiley face and Volkan's hilarious opposition in the room provided great felicity for me while working. I also would like to thank my dear friend and colleague Kerem Demirtaş, whose great technical support and friendship was also a great asset for me. I also want to offer my regards to my dear other colleagues Tülin İnkaya, Ayşegül Demirtaş, Banu Lokman, Aykut Bulut and Çınar Kılcıoğlu as their presence and company were really relaxing for me.

I also would like to express my special thanks to the special person in my life; Ayla Öylek for being with me; giving her love and making me feel better in every instance during this study. I am extremely fortunate for having her love and support at the most hopeless moments. Without her, completing this work in such a happy mood would not be possible.

And, lastly I would like to thank all the members of the examining committee and all the people I had the chance to know during the period of my assistantship in the Department of Industrial Engineering.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURES

LIST OF TABLES

TABLES

CHAPTER 1

INTRODUCTION

Project is a set of interrelated activities undertaken to create a unique product or service. Project management deals with planning, organizing and successful completion of the project. Project scheduling defines the start and completion times of the activities, together with resource assignments; hence it is essential for the successful completion of the project. Project management finds its applications in many areas, including but not limited to, software development, new product designs, construction industry and make-to order production environments when each customer order defines a unique output.

When the only feature associated with project activities is duration, the objective of project scheduling is simply to minimize the project completion time. The minimum completion time is found in polynomial time with Critical Path Method *(CPM)* when activity durations are deterministic and with Program Evaluation Review Technique *(PERT)* when they are probabilistic.

In many practical applications, the activities in a project consume resources other than time, such as money, labor, equipment and machine. When these resources are scarce, the problem becomes Resource Constrained Project Scheduling Problem *(RCPSP)*. The *RCPSP* is the most common project scheduling problem

encountered in Operational Research literature. The objective is to minimize the project completion time subject to the resource constraints.

The resources used in the *RCPSP* are of three types: Renewable resources, non-renewable resources and doubly constrained resources. A renewable resource is available at a certain amount for all periods whereas a non-renewable resource is a fixed amount declining through usage. Some resources are both renewable and non renewable and called doubly constrained.

When the resources are scarce, their planning and organization are of crucial importance. Allocating the resources to the correct activities on correct times results in good implementable schedules. Resource leveling and resource investment problems are two main problem types that are pertinent to resource based objectives. The resource leveling problems smooth the resource consumptions among all periods. The resource investment problem minimizes the relevant costs associated with the resources.

A class of the project scheduling problems, so-called multi-mode problems, is characterized by their activity modes. An activity has more than one mode if it has processing alternatives, for example its duration can be reduced by putting extra resources. Each mode of an activity has two main components: time and resource consumption. The associated problem is referred to as time/cost trade-off problems if the resource used is money and money is the nonrenewable resource. Time/Cost trade-off problem deals with the trade-off between the project completion time and the cost associated with it. Another trade-off is between project completion time and the renewable resource consumption. Increasing the maximum possible resource available for each period, the project completion time may decrease and vice-versa. This problem is called the time/resource trade-off problem.

In this study we consider the resource investment problem with time/resource trade-offs. Our problem consists of a single renewable resource, like the workers in a construction project, and given a project deadline which may be defined by contractual agreements. The objective is to minimize the maximum resource consumption over all periods. The maximum requirement defines the amount of resources used, when all units of the resource are made available at the beginning of

the project and are freed altogether when the project is complete. For example the number of construction workers hired for a construction project may be defined by the period that requires maximum number of workers. In such a case the total labor investment is an increasing function of the number of the workers. The number of available workers allocated to a project can be reduced selecting an appropriate execution mode for each activity.

We, in this study, develop powerful problem size reduction mechanisms, lower bounding and heuristic procedures for the resource investment problem with time/resource trade-offs. Despite its practical importance, the research on the problem is quite limited. To the best of our knowledge, there is only one study that deals with multiple renewable resources and proposes a priority based heuristic to solve the problem. We hope that our study fills an open area in research constrained project scheduling literature.

The rest of the thesis is organized as follows: In Chapter 3, we give a literature review for the related resource constrained project scheduling problems. In Chapter 4, we define our problem, give a mathematical model and discuss mode elimination techniques. Chapter 5 discusses our solution approaches; lower bounds and heuristics. Chapter 6 reports on our computational experiments and discusses its results. In Chapter 7 we give a brief conclusion of our study.

CHAPTER 2

PROJECT SCHEDULING

In this chapter we first define a project and then present the Activity on Node and Activity on Arc project networks. We present single mode problems and their solutions and support our discussion with an example. We then introduce the Resource Constrained Project Scheduling Problem and multi mode Time/Cost Trade-off and Time/Resource Trade-off problems.

2.1 Project in General

A project is a "*temporary endeavor undertaken to create a unique product or service"* according to the Project Management Institute. A project is a set of activities which are interrelated and which share resources such as money, labor, machine, equipment.

An activity is the smallest work element of a project. The activities are interrelated because they have an input-output relation. Each activity has predecessors and successors. An activity, say activity *i*, is a predecessor of another activity, say activity *j*, if it has to finish before *j* starts. Due to the transitivity property of precedence relations, that is if *i* precedes *j* and *j* precedes *k* then, *i* also precedes *k*, there are many redundant expressions. An easy way of expressing all the precedence relations without any redundancy is using the immediate predecessor relation. If activity *j* can start immediately after activity *i* finishes, then activity *i* is

an immediate predecessor of activity *j*.The relations between the activities can be represented using many tools, the most common ones being Gantt Chart or Project Networks.

Gantt Chart

A Gantt chart is a bar chart that is developed by Henry L. Gantt in early 1900s first as a production control tool. A Gantt Chart, provides valuable information about the project such as the start and completion times of activities, precedence relations, activities being processed in parallel. However, it does not provide insight on the criticality of activities. An example of a Gantt Chart can be seen in Figure 2.1.

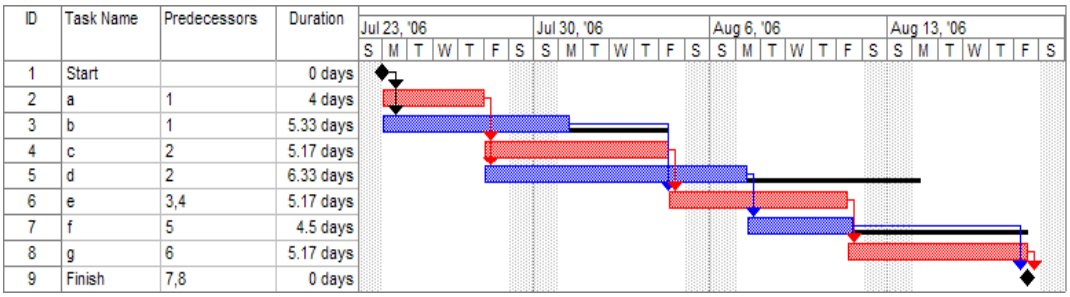

Figure 2.1 – A Gantt Chart with 7 activities

2.2 Project Networks

Project Networks illustrate the activities and their relations on a network. They provide valuable information on the criticality of the activities, and their allowances.

The activities of a network can be represented in two different schemes.

1. Activity on Arc *(AoA)* representation
2. Activity on Node *(AoN)* representation

2.2.1 *AoA* Representation

In *AoA* networks, an arc represents an activity and a node represents an event. In general, an event corresponds to the start and/or completion time of an activity, it may also correspond to a particular milestone for the project, like the half completion of project, the entire completion of the project. *AoA* representation may

require dummy arcs or dummy activities for proper representation of the precedence relations.

We use one instance adapted from our problem set (we call *Example problem* hereafter) to illustrate the *AoA* Network. Table 2-1 tabulates the immediate precedence relations and Figure 2.2 illustrates these relations on an *AoA* network. Note that activities 1 and 12 are dummy activities that define the starting and ending of the project.

Table 2-1 – The precedence relations for the example problem

Activity	Immediate Predecessors
1	- (Starting Dummy Activity)
2	1
3	1
4	1
5	2
6	4
7	4
8	2,4
9	4
10	5,8
11	3,6,7,9
12	10,11(Ending Dummy Activity)

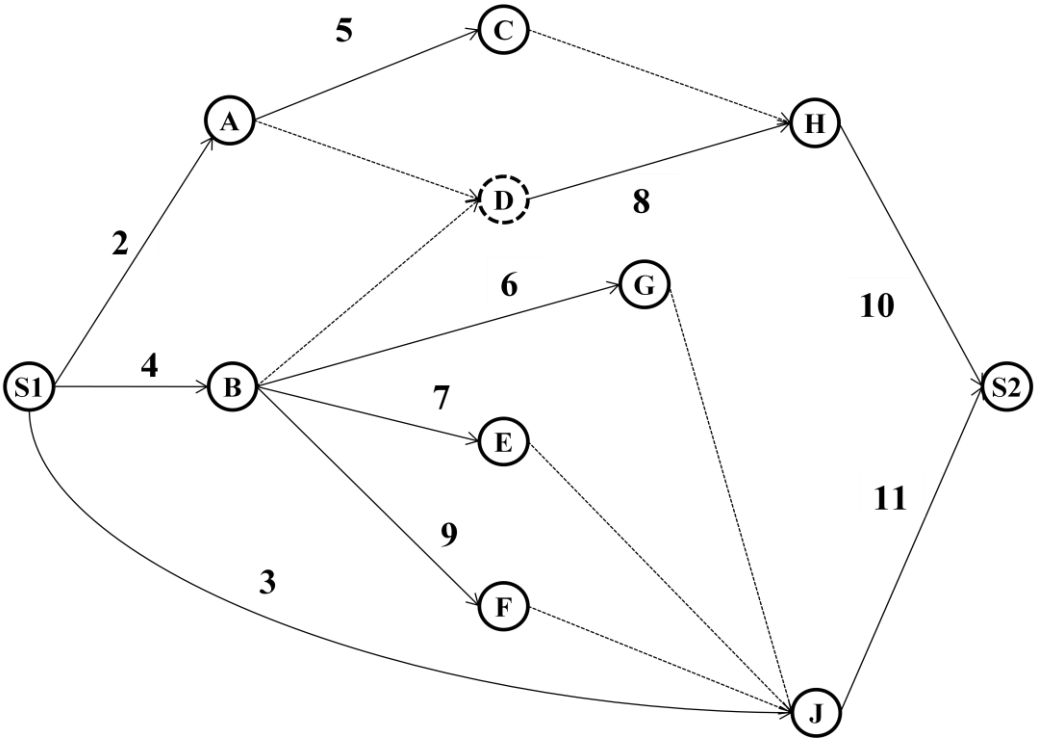

Figure 2.2 – *AoA* network for the example problem

In the project network in Figure 2.2- each arc corresponds to an activity and each node corresponds to an event. Nodes S1 and S2 represent start and finish of the project respectively. Each node corresponds to an event; node J, for instance, is for the completion of Activity 3 and three dummy activities and start of activity 11. Each arc is an activity; the arc between nodes B and E represents the Activity 7. The dummy arcs are for reflecting the precedence relation properly. The arc between C and H is a dummy arc to illustrate the precedence relation between activities 5 and 10.

2.2.2 *AoN* Representation

In *AoN* networks, the nodes represent activities and the arcs represent the immediate precedence relations. Activity *i* is an immediate predecessor of activity *j* if an arc is directed from *i* to *j*. Node 1 is the starting activity that consumes no time and Node 12 is the ending activity whose duration is zero. Figure 2.3 displays our *example problem* on an *AoN* network.

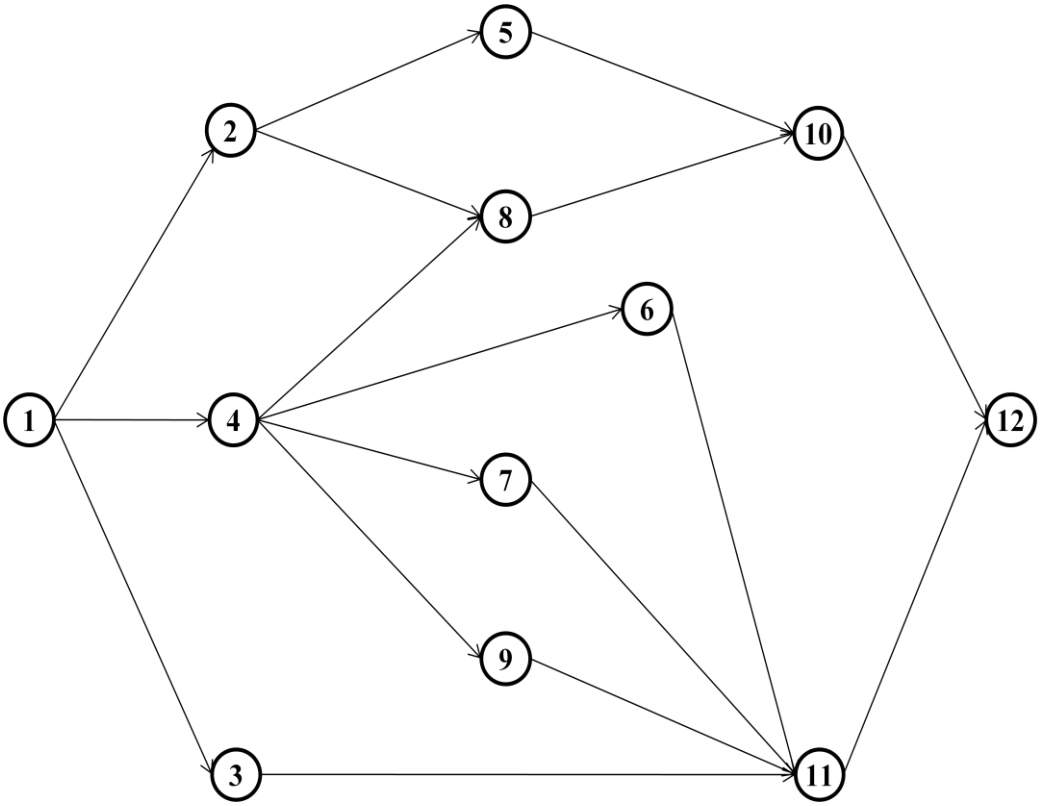

Figure 2.3 – *AoN* network for the example problem

In Figure 2.3, note that activity 1 is the node that represents the start of the project and activity 12 is the node where the project finishes.

2.3 Project Scheduling

Project Scheduling decides on the start and finish times of the activities, and the allocation of all scarce resources to the activities.

All activities in a project have a processing time and resource consumption. Labor, machines, and money are some instances for the resources. In general when the resource of interest is renewable, it might be considered as labor and when the resource is non-renewable the most common indicator of the resource is money. Sometimes, these resources can be consumed all together. In our problem, the single resource is renewable.

If all activities in a project have only one combination of duration and resource consumption, the project is considered as a single-mode project. When there are multiple alternative ways of processing an activity, the project is considered as a

multi-mode project. Each alternative way, called the mode of activities, has its own processing times and resource usages.

2.3.1 Single-Mode Project Scheduling Problems

For the single mode project scheduling problems, the purpose is to assign a start time to all the activities so that the precedence relations are respected and the project is completed in the earliest possible time. The well known Critical Path Method *(CPM)* is used to find such a schedule. We first explain *CPM*, that will be referred often throughout the thesis. For detailed discussion on *CPM*, see Meredith and Mantel (2003).

The Critical Path Method

The *CPM* is a method of finding the critical path, its length (earliest possible completion time of a project), critical activities and earliest and latest start times for the activities. We need following definitions to explain the method.

Critical Path: The longest path(s) in a project. The length of the critical path defines the project completion time.

Critical activities: Any activity in any critical path is a critical activity. The main property of a critical activity is that, its earliest start time is equal to its latest start time hence if the activity starts later than its earliest start time; the schedule cannot meet on time.

Noncritical Activities: Any activity for which the latest start time is greater than its earliest start, i.e., any activity on any critical path.

Total Slack: The difference between earliest and latest start times of an activity defines its total slack. Total slack is the amount an activity can be delayed without affecting the project completion time. So, the total slack of a critical activity equals zero.

While starting the procedure, the earliest start times of activities with no predecessor (whose predecessor is the starting dummy activity) are set to 0 and their earliest completion times are calculated. The earliest start time of any other activity is equal to the maximum of its predecessors' earliest completion time. The ending dummy activity's earliest completion time (recall that its processing time is 0) is the

project's earliest completion time. Then, this earliest completion time becomes the latest completion time for the activities which have no successors (whose only successor is the ending dummy activity). Using the durations of activities, late start times for these activities are calculated. For all the other activities, the latest completion time is equal to the minimum of its immediate successors' latest start times. When the earliest and latest possible start times are known for any activity, total slack times and criticality of all the activities can be defined.

The notation, which is needed for *CPM*, and used throughout this thesis, is given below:

p_{ij}*:* processing time of activity *i* at its mode *j*.

E_i*:* Set of immediate predecessors of activity *i.*

$Succ_i$*:* Set of immediate successors of activity *i*.

ES_i: Earliest start time for activity *i*.

LS_i: Latest start time for activity *i*.

EC_i: Earliest completion time for activity *i*.

LC_i: Latest completion time for activity *i*.

CR: Set of critical activities

$Slack_i$: Total slack of activity *i*.

When the problem is single mode, we modify our parameter p_{ij} as p_i.

Using the notation, we formally define the algorithm; which is excerpted from Değirmenci (2008);

Initialization:

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

Main Body:

Repeat

$$ES_i = Max_{k \in E_i} \; ES_k + p_k \qquad i : \forall k \in E_i \; ES_j \text{ is calculated}$$

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

$$T = Max_i \; ES_i + p_i$$

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

Repeat

$$LC_i = Min_{k \in S_i} \; LC_j - p_j \qquad i : \forall j \in Succ_i \; LC_j \text{ is calculated}$$

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

Finalization:

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

The method will be illustrated in the example problem. Note that our problem is a multi-mode problem, so we choose the shortest duration modes for each activity to illustrate *CPM* and exclude resource consumption as it does not affect *CPM* calculations. The activity information is presented in Table 2-2.

Table 2-2 – The precedence relations and durations for the example problem

Activity	Immediate Predecessors	Duration (days)
1	- (Starting Dummy Activity)	-
2	1	7
3	1	9
4	1	8
5	2	8
6	4	6
7	4	8
8	2,4	9
9	4	7
10	5,8	6
11	3,6,7,9	9
12	10,11(Ending Dummy Activity)	-

Now, step by step we show the calculations,

Initialization:

$E_2 = E_3 = E_4 = \varnothing$ (The only predecessor is the starting dummy activity); $ES_2 = ES_3 = ES_4 = 0$

Main Body:

$E_5 = 2$ $\quad ES_5 = ES_2 + p_2 = 7$

$E_6 = \{4\}$ $\quad ES_6 = ES_4 + p_4 = 8$

$E_7 = \{4\}$ $\quad ES_7 = ES_4 + p_4 = 8$

$E_8 = \{2,4\}$ $\quad ES_8 = Max\{ES_2 + p_2, ES_4 + p_4\} = 8$

$E_9 = \{4\}$ $\quad ES_9 = ES_4 + p_4 = 8$

$E_{10} = \{5,8\}$ $\quad ES_{10} = Max\{ES_5 + p_5, ES_8 + p_8\} = 17$

$E_{11} = \{3,6,7,9\}$ $ES_{11} = Max\{ES_3 + p_3, ES_6 + p_6, ES_7 + p_7, ES_9 + p_9\} = 16$

$T = Max\{ES_{10} + p_{10}, ES_{11} + p_{11}\} = 26$

So, as T is found now, $T = 26$, we are ready to calculate the latest completion times.

$Succ_{10} = Succ_{11} = \varnothing$ (The only successor is the ending dummy activity); $LC_{10} = LC_{11} = T = 25$

$Succ_9 = \{11\}$ $LC_9 = LC_{11} - p_{11} = 16$

$Succ_7 = \{11\}$ $LC_7 = LC_{11} - p_{11} = 16$

$Succ_6 = \{11\}$ $LC_6 = LC_{11} - p_{11} = 16$

$Succ_3 = \{11\}$ $LC_3 = LC_{11} - p_{11} = 16$

$Succ_8 = \{10\}$ $LC_8 = LC_{10} - p_{10} = 19$

$Succ_5 = \{10\}$ $LC_5 = LC_{10} - p_{10} = 19$

$Succ_4 = \{6,7,8,9\}$ $LC_4 = Min\ \ LC_9 - p_9, LC_8 - p_8, LC_7 - p_7, LC_6 - p_6\ \ = 8$

$Succ_2 = \{5,8\}$ $LC_4 = Min\ \ LC_8 - p_8, LC_5 - p_5\ \ = 10$

After calculating earliest start and latest completion times, we finalize the algorithm by specifying earliest completion times, latest start times and slacks for each activity. Together with immediate predecessors, activity durations, earliest start and latest completion times, these values are tabulated in Table 2-3.

Table 2-3 – The early start-late start times and slack values for activities

Activity	**Immediate Predecessors**	**Duration (days)**	ES_i	EC_i	LS_i	LC_i	$Slack_i$
1 (dummy)	-	-	0	0	0	0	0
2	1	7	0	7	3	10	3
3	1	9	0	9	7	16	7
4	1	8	0	8	0	8	0
5	2	8	7	15	11	19	4
6	4	6	8	14	10	16	2
7	4	8	8	16	8	16	0
8	2,4	9	8	17	10	19	2
9	4	7	8	15	9	16	1
10	5,8	6	17	23	19	25	2
11	3,6,7,9	9	16	25	16	25	0
12 (Dummy)	10,11	-	25	25	25	25	0

The activities with positive slack values, called noncritical activities, can be delayed by their slack values without affecting the project completion time. For instance, activity 3 can start 7 days after its earliest start time without changing the project completion time. As their slack values are equal to 0, activities 4, 7 and 11 are the critical activities.

Figure 2.4 illustrates the network with earliest start and latest completion times in boxes and duration of activity in parentheses next to the associated node with the critical activities and path is bold.

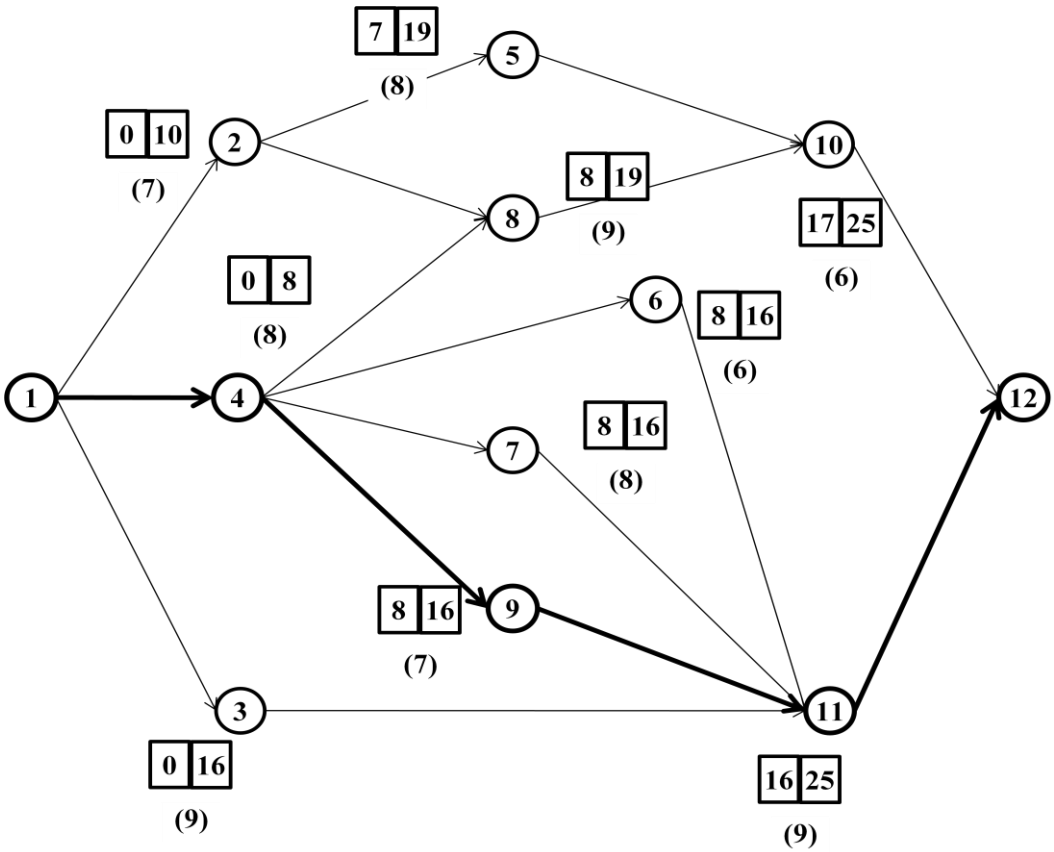

Figure 2.4 – The *AoN* network for the example problem with early/late start times and durations.

Resource Constrained Project Scheduling Problems: A single mode project scheduling problem, when the resources of interest are scarce, is called the Resource Constrained Project Scheduling Problem *(RCPSP)*. The resource constrained project scheduling problem can be of two types: resource allocation and resource leveling. The resource allocation problem aims to minimize the project completion time given the resource requirements for each activity and a limit on each resource. So the resource allocation problem allocates the scarce resources to activities without exceeding the resource capacities and by completing the project early as possible. The resource leveling problem tries to balance the resource usages throughout the defined life of the project.

2.3.2 Multi-Mode Project Scheduling Problems

Sometimes it is possible that an activity may be executed in different ways. These ways are called modes in the project scheduling problem. This stems from the fact that allocating more resources to an activity may shorten its duration. Considering that the single resource is non-renewable, a simple example is that, while a job can be done in 3 hours spending 5 units of money, or can be done in 5 hours spending only 1 unit of money. When the single resource is renewable, the idea is the same, for instance; when 5 workers are assigned to an activity it is completed in 6 days, whereas when 4 workers are assigned to that activity it finishes in 7 days.

The multi-mode project scheduling problems have an additional complexity brought by mode selection decisions. When there is a single resource non-renewable resource, the problem is called the Time/Cost Trade-off Problem, and when the resource is renewable it is Time/Resource Trade-off Problem.

The time/cost and time/resource trade-off problems are of two types: continuous and discrete. In continuous problems, activity time is any increasing function of its resource consumption. In continuous trade-off problems there are infinitely many modes defined by the continuous time/cost function. In discrete problems, there are specified number modes where one of which is selected for each activity. In this study, we consider discrete time/resource trade-off problems.

The Discrete Time/Cost Trade-off Problem

The Discrete Time/Cost Trade-off Problems can be of three types: deadline, budget and curve problems. The Deadline Problem minimizes the total resource consumption cost subject to a predetermined project deadline. The Budget Problem minimizes the project completion time when the budget is limited. The Time/Cost Trade-off Curve Problem has two objectives, total project cost and project completion time. The problem is to generate all efficient, i.e. non-dominated, solutions with respect to these criteria.

The Discrete Time/Resource Trade-off Problem

The aim of the Discrete Time/Resource Trade-off Problem *(DTRP)* is to find a schedule for each activity, with a single renewable resource, in one of its defined modes that minimizes the project completion time subject to precedence constraints and resource constraints according to Ranjbar and Kianfar (2007). A mode is composed of a processing time and resource requirement and the activities have many execution modes and one of them must be chosen.

With the difference of the objective function, our problem has the same structure. In our problem, we try to minimize the capacity of our single non-renewable resource without violating the project completion time constraint. Project completion time is the length of the critical path calculated using the smallest modes of activities. The data for modes are given in Table 2-4, p_{ij} denoting the processing time for mode j of activity i and r_{ij} denoting the resource requirement of mode j of activity i.

Table 2-4 – The execution modes for non-dummy activities

ACTIVITIES		MODES																			
	i	p_{i1}	r_{i1}	p_{i2}	r_{i2}	p_{i3}	r_{i3}	p_{i4}	r_{i4}	p_{i5}	r_{i5}	p_{i6}	r_{i6}	p_{i7}	r_{i7}	p_{i8}	r_{i8}	p_{i9}	r_{i9}	p_{i10}	r_{i10}
	2	7	8	9	6	11	5	13	4	18	3	27	2	55	1	-	-	-	-	-	-
	3	9	11	10	9	11	8	13	7	15	6	19	5	23	4	31	3	47	2	95	1
	4	8	8	9	7	10	6	12	5	16	4	21	3	32	2	64	1	-	-	-	-
	5	8	10	9	8	10	7	12	6	14	5	18	4	24	3	37	2	74	1	-	-
	6	6	8	7	6	9	5	11	4	15	3	22	2	45	1	-	-	-	-	-	-
	7	8	9	9	7	11	6	13	5	16	4	22	3	33	2	66	1	-	-	-	-
	8	9	11	10	9	12	8	13	7	16	6	19	5	24	4	32	3	48	2	97	1
	9	7	9	8	7	9	6	11	5	14	4	19	3	29	2	59	1	-	-	-	-
	10	6	8	7	6	8	5	10	4	14	3	21	2	43	1	-	-	-	-	-	-
	11	9	9	10	8	11	7	13	6	16	5	20	4	27	3	40	2	81	1	-	-

The critical path for a multi mode problem scheduling problem is calculated using the modes with smallest processing times. That was what we have done in *CPM* calculations. As we use the smallest processing time modes in *CPM* calculations, the length of the critical path is 25 time units. To gain insight to the problem a schedule, namely the earliest start schedule is constructed and the resource profile of it is presented in Figure 2.5.

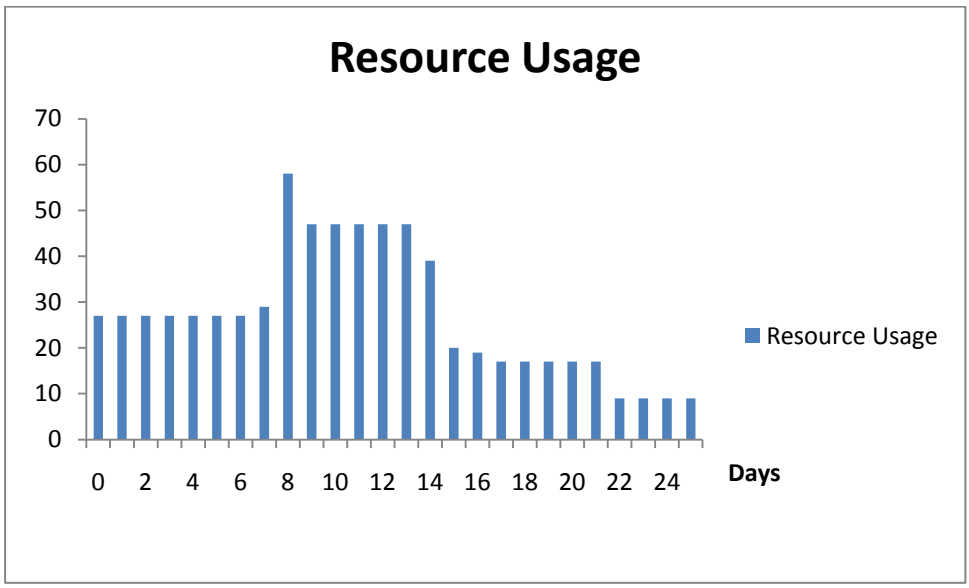

Figure 2.5 – The resource usage profile of early start schedule.

Maximum resource usage is 58 units in this project when all activities start at their earliest possible times. As Figure 2.5 illustrates the resource profile is not very good. This schedule may get better, in the rest of the thesis, we try to find an efficient way to obtain better results.

CHAPTER 3

LITERATURE REVIEW ON RESOURCE CONSTRAINED PROJECT SCHEDULING PROBLEMS

In this chapter, we review the project scheduling problems that are most closely related to our resource investment problem with time/resource trade-offs. Section 3.1 and Section 3.2 overview the literature on the makespan problem with time/resource trade-offs and resource investment problem with single and multi modes, respectively. For other resource constrained project scheduling problems, we refer the reader to the review papers by Herrolelen, De Reyck and Demeulemester (1998) and Hartmann and Briskorn (2010) for the detailed information

3.1 Makespan Problem with Time/Resource Trade-Offs

Multi mode resource constrained problem assumes several alternatives for time versus resource consumption. If these alternatives are specified at discrete time points then the problem is referred to as discrete time/resource trade-off project scheduling problem *(DTRP)*. If the time/resource trade-off function receives values at all continuous values between specified limits, the associated problem is continuous time/resource trade-off project scheduling problem.

The majority of the *DTRP*s consider the project completion time, so-called makespan, as the objective function. The other objectives considered in the literature are resource related like resource consumption cost and resource leveling There exist several studies that solve the *DTRP* problem so as to minimize the makespan. We classify the *DTRP* studies into two groups as optimization and approximation studies.

3.1.1 Optimization Studies

Talbot (1982), Sprecher et al.(1997) and Demeulemeester et al. (1999) propose branch and bound algorithms to find exact solutions. Talbot (1982) defines and formulates the discrete time resource trade-off makespan problem. He considers both renewable and non-renewable resources and nonpreemptive activities. He proposes a two stage algorithm. In the first stage, a re-labeling procedure is applied to the activities, modes and resources. The relabeling is assigning priorities to these elements of the problem that will be used while enumerating in the second stage. In the second stage the algorithm tries to assign the jobs sequentially and feasibly. Upon finding a good solution, bounds are tightened and enumeration is continued. An optimal solution is found definitely when either the enumeration process is completed or makespan value equal to a known lower bound is reached. His computational results reveal that optimal solutions for small-sized problems can be found easily. For large-sized instances, the procedures provide high quality approximate solutions.

Sprecher et al.(1997) propose a branch and bound algorithm for an exact solution of the problem. Their enumeration tree resembles the precedence diagram. The activities that are eligible (whose predecessors are processed) to start are branched from a mother node. The first feasible mode is chosen when an activity is being scheduled. If all modes are infeasible then the procedure backtracks. To reduce the search tree nine rules are proposed. Some of these rules are static and applied at the beginning of the procedure and some of them are dynamic and applied during the procedure. However, they observe that some of those reduction rules are not worth applying as the time to test them outweighs the savings obtained. The algorithm is tested using instances generated by *ProGen* that is available in *PSPLIB.* In the tests different combinations of the reduction rules are used, i.e. all search tree reduction

rules are not used all together. All instances up to 20 activities can be solved within 10 seconds.

Demeulemeester et al. (1999) present a branch and bound procedure for the discrete time resource problem with nonpreemptive activities and a single renewable resource. They propose four dominance rules to eliminate the activity modes. The partial schedules are constructed by temporarily scheduling all feasible nondominated maximal activity-mode combinations. An activity-mode combination is feasible if the resource constraint is not violated; maximal if no other activity can be executed in parallel in any of its modes and nondominated if it cannot be dominated by their dominance rules. The decision point in the branching scheme is the nearest activity finishing time, t. The activities that are in progress t can be removed from the partial schedule or they can be rescheduled, but the activities that are completed at time t, cannot be removed from the partial schedule or rescheduled. Two lower bounds are calculated and their maximum is used in the algorithm. The first one ignores the resource constraint and hence based on the critical path. The critical path starting with the eligible (either started at the decision point or became eligible as its predecessor is completed) activity is calculated and added to the next decision point. The second is a resource based lower bound; total work content is divided to the each period's resource availability of the renewable resource and rounded up to the nearest integer. Demeulmeester et al. generated their own problem sets and solved them to optimality in less than 1000 seconds mostly. All the problems with up to 4 modes and 30 activities are solved to optimality within a maximum CPU time of 160.54 seconds.

3.1.2 Approximation Studies:

The majority of the approximation studies on the *DTRP* is on the Genetic Algorithms. Tabu Search Algorithm, Simulated Annealing Algorithm and Scatter Search Algorithm are used each in a single paper.

Tabu Search Algorithms: De Reyck et al. (1998), consider a single renewable resource and nonpreemptive activities. They propose and compare steepest descent, fastest descent, iterated descent, randomized search and tabu search algorithms. They define the neighborhood structure as changing only one of the activities' execution mode. The search prohibits recently made moves; it allows revisiting a

solution but only with changing the direction of the search. The generated instances are used for testing the algorithms and comparing them. They also compare the local search algorithms with the branch and bound algorithm proposed by Demeulmeester et al. (1999). Random procedure being the worst performer in terms of solution quality, Tabu search seems to be the slowest in small instances although it is the one which deviates the least from the best solution among the local search methods. Its performance in larger instances is far better than all other local search methods when both time and quality of the solution are considered.

Genetic Algorithms: Alcaraz et al. (2003) propose a genetic algorithm for the renewable and nonrenewable resources and nonpreemptive activities. They first reduce the problem size by applying the preprocessing rules introduced by Sprecher et al (1997) and used also by Hartmann (2001) are applied. The preprocessing rules eliminate inefficient modes, non-executable modes and redundant non-renewable resources. They solve the reduced problem by a genetic algorithm. Their algorithm uses a chromosome representation, such that an individual is represented by I=(λ,f/b,μ) where λ represents an ordered activity list, *f/b* indicates the scheduling generation scheme used to build the schedule, serial forward or backward and μ is the mode assignment. They allow the infeasible solutions with respect to only non renewable resources as finding feasible solutions with more than one non-renewable resource is NP-complete. They evaluate the fitness values by the objective function values and feasibility of the solutions. Test problems are gathered from the PSPLIB and these instances are compared with the best results of the previously published studies. It is seen that deviation from optimal solution is very small and much better than any other reported heuristic. The algorithm is also so fast that any problem instance is solved in less than a second before it reaching a stopping condition.

Ranjbar and Kianfar (2007) consider renewable resources and nonpreemptive activities. They propose a genetic algorithm, using a very simple representation scheme. They keep the priorities for the activities in one array and in another array the mode assignments of corresponding activities are held. While assigning the priorities to the activities, a topological ordering is used. This ordering is related to the precedence relations of the activities and introduced by Valls et al. (2003) is used. A Schedule Generation Scheme is used to assign activities with priorities

specified in the schedule representation. The population is formed using a diversification generation method which keeps track of the frequency of the priority assignments and gives less chance to the frequently appearing solution elements. The survival of the fittest is provided by assigning deletion probabilities to the members of population directly proportional to the makespan. They compare their algorithm with that of De Reyck et al. (1998) and show its outperforming performance.

Mori and Tseng (1997) propose a genetic algorithm for a single renewable resource and nonpreemptive activities problem. A scheduling order for the activities is defined. This order has two components; a forward order and a backward one. These orders are defined using the precedence information. The fitness values are calculated using the duration of the activity. The crossover operator uses one random parent and the best current solution. It specifies a junction activity and takes activities scheduled up to that activity from one random parent and remaining from the other one. There are two mutation schemes, one changing modes and other creating a brand new schedule. The algorithm is tested on the instances generated by the authors. They compared their results with those of Drexl and Gruenewald (1993) by using their own instances.

In another effort to solve the problem with genetic algorithms Peteghem and Vanhoucke (2009) study both the preemptive and non-preemptive activities. The mode elimination procedure proposed by Sprecher et al. (1997) is used for preprocessing. The representation scheme is identical with Ranjbar and Kianfar (2007). To assign the activities in the priority list to a schedule, the serial schedule generation scheme introduced by Kelley (1963) is improved and used. Two extensions to this scheme are discussed. One is the forward/backward scheduling technique that makes use of justifying the activities to right and left. The other one is a mode improvement procedure, with some probability mode improvement procedure is applied to an activity, when there is an improvement, the schedule is changed. For initial population left justified schedules are taken regardless of their feasibility. The infeasibilities are tried to be resolved by a local search procedure. If they cannot be resolved after some steps; the infeasible solutions are allowed to stay in the population with a penalty associated with them in the fitness value. The

penalties are calculated using makespan values. For crossover an activity is selected randomly and a one point crossover is applied. The algorithm is compared with other heuristics using PSPLIB and Boctor instances. The results indicate that the algorithm performs well in terms of both solution quality and computational time.

Simulated Annealing Algorithms: Jozefowska et al. (2001) consider nonpreemptive activities and renewable resources. The objective is minimizing the makespan. Simulated Annealing *(SA)* is the metaheuristic used in the study. The representation consists of a list of activities and a list of execution modes. Two different approaches are used. The first does not permit infeasibility whereas the second one permits infeasibility with a penalty. In both approaches, the neighborhood structure is defined as randomly selecting an activity, moving it and its immediate predecessors and successors within their allowable range or changing the mode of the activity to a random mode or doing both together. Jozefowska et al. (2001) use *PSPLIB* instances, in small instances *SA* does not perform well enough. Only 35-40% of the instances with 10 to 20 activities can be solved to optimality. About 55.8% of the instances with 30 activities *SA* finds the best known solution.

Scatter Search Algorithms: Ranjbar et al. (2009) consider renewable resources and nonpreemptive activities. They use the same schedule generation scheme with Ranjbar and Kianfar (2007) and study on a scatter search. The initial population is formed with the same method but an intensification phase using a local search is used for a better initial population. Then, two diversified reference sets are formed using a distance based measure. Path relinking is used to combine solutions in reference sets. Instances from the *PSPLIB* are used to test the algorithm and scatter search is found to be the best performing metaheuristic. Branch and bound algorithm of Demeulmeester et al. (2008) fails to present an optimal solution for many large-sized instances for which scatter search returns high quality solutions using same CPU time.

3.2 Resource Investment Problems

The resource planning problems in project scheduling can be divided into two classes: the resource leveling and resource allocation. The leveling problem occurs, when sufficient resources are available and one tries to keep the resource usage at a constant rate as much as possible. The resource leveling problems have two classes:

minimizing the range of resource usage and minimizing cost of resource usages. The cost of resource usages problem is referred to as resource investment problem or resource availability cost problem. The resource allocation problem occurs when the total resource usage is restricted and the objective is to allocate various resources to the activities so as to minimize the project completion time.

3.2.1 Single Mode Resource Investment Problem

Möhring (1984) shows that, the resource investment problem with non-preemptive tasks and a single renewable resource is strongly NP-hard. Möhring (1984), Ranjbar et al.(2009), Radermacher (1978), Drexl and Kimms (2001), Demeulemeester (1995), Yamashita et al.(2004), Shadrokh and Kianfar (2007) and Ranjbar et al.(2009), study the resource investment problem with non-preemptive tasks and renewable resources.

Möhring (1984) uses the concept of feasible partial orders, defined by Radermacher (1978) to solve the problem. He extends the precedence relation of initial network respecting the resource limits and the time limit and obtains feasible partial orders. The set of these feasible partial orders lead to the optimal solution. He also defines a duality relation between two problems.

Drexl and Kimms (2001) propose two lower bounds and two optimization guided heuristics that are byproducts of their lower bounds. The first lower bound makes use of Lagrangian Relaxation technique and divides the problem into two subproblems that can be solved in pseudo- polynomial time. A good side effect of the approach is that one of their subproblems yields a feasible schedule. They run the procedure for a certain number of iterations and select the best lower and upper bounds. The second lower bound is based on a column generation technique in which each column represents a feasible schedule, hence an upper bound. To obtain a lower bound, they solve the LP relaxation of the master problem, and solve the dual of the LP model to find out if any column is to be added to the problem. Their computational experiment reveals that the quality of the lower and upper bounds depend on the deadline value and the number of resources, but not on the complexity of the precedence relations.

Demeulemeester (1995) proposes a branch and bound method for the non-decreasing resource cost function problem. He defines the efficient points based on their resource availabilities. A point, call i, having resource availabilities $(a_1,..,a_m)$ is efficient if no point j having resource availabilities $(a_1',..,a_m')$ exists such that all $a_k' \leq a_k$ for $k=1,..m$. At all branches a resource-constrained project scheduling decision problem is solved. The strategy is to solve for the cheapest efficient point at all stages, if a feasible solution is found, then the solution is optimal. If the decision problem returns an infeasible solution, then the point is removed from the solution space and new efficient points are added to the efficient set and the procedure continues until a feasible solution is found. He compares his computation times with those of Möhring's and finds out that his algorithm is approximately 100 times faster. He also compares his results with those of Patterson's Resource Constrained Project Scheduling Problem and finds that his algorithm uses about 5% lower resource for the same makespan.

Yamashita et al.(2004) propose a scatter search algorithm which uses an improvement heuristic proposed by Tormos and Lova.(2004). In the reference set, best solutions are accompanied with diverse solutions. They use various methods for diversification of the reference set such as frequency based memories and path relinking. They select activity pairs from the reference set and apply a combination method to obtain some new solutions. The procedure continues until a previously specified number of solutions are checked. They compare their results with two simple heuristics each involving an improvement step. Their algorithm outperforms the heuristics in terms of quality, and its computational time is slightly higher.

Shadrokh and Kianfar (2007) consider the problem of minimizing the total resource investment and tardiness penalty. They propose a genetic algorithm where the chromosomes are represented in two parts, one denoting the activity list, the other capacity list. These lists are transformed to schedules by using Serial Schedule Generation Scheme and Parallel Schedule Generation Scheme. With certain probabilities, the chromosomes are subjected to a crossover or a mutation. After each operation, the chromosome is subjected to a local search and at the time that the population will be renewed with new solutions, an immigrant is added to the

population with a certain probability. We find that their algorithm finds optimal solutions for many problems with up to 20 tasks.

Ranjbar et al.(2009), use the activity list representation to represent the priority structure between the activities. In the activity list, the activity with the highest priority stands in the first position next to the dummy starting activity. Using early start and late start schedules they move the activities between their allowable ranges until no improvement is possible. Using solutions generated by a biased random sampling a reference set is formed and pairs in the reference set are subjected to a path relinking procedure. This procedure creates as many solutions as required.

Ranjbar et al.(2009), also propose a genetic algorithm using the activity lists as the representation scheme and obtain new solutions using two point crossover and mutation operators. The termination condition is the same as that of path relinking. They compare their results with those of Yamashita et al.(2004) and show that the path relinking algorithm outperforms their genetic algorithm.

3.2.2 Multi Mode (Time/Resource Trade-Off) Resource Investment Problem

Hsu and Kim (2005), Talbot (1982), Nudtasomboon and Randhawa (1997), Skarmeta et al. (1999) and Pulat and Horn (1996) study time/resource trade-off problem with resource cost criteria. All time/resource trade-off resource scheduling studies assume discrete alternatives, with one exception. The exception is due to Pulat and Horn (1996) that considers continuous version of the problem. We classify the studies into two groups as single criterion and multi criteria studies.

Single Criterion Problems: Hsu and Kim (2005) consider the multi mode resource investment problem for nonpreemptive activities consuming renewable resources. They calculate the priority function value for each available activity, its mode, and starting time by combining two functions. One of the functions calculates the impact of the decision to the unscheduled activities using the slacks. The other function calculates the resources consumed by the alternative activity-mode-time combination. At each decision point they consider all activities whose predecessors are completed and choose the one having smallest priority function value. They compare their heuristic with some other priority rule based heuristics. They show that their heuristic outperforms these rules.

Talbot (1982) extends his solution methodology for the makespan objective to the minimum resource usage problem subject to the specified project completion time value.

Multi Criteria Problems: Nudtasomboon and Randhawa (1997) extend the idea of Talbot (1982) and propose some improvements to his algorithm for some other objectives and problem characteristics. They use Talbot's enumeration scheme and propose some mode elimination rules and bounding schemes. Three objectives makespan, total cost and resource leveling, are discussed and a priority based multi-objective solution procedure is proposed. For each objective, specific labeling and backtracking rules are proposed. Their computational results with 18 problem instances reveal the superiority of their algorithms over that of Talbot's.

Skarmeta et al. (1999) consider both renewable and nonrenewable resources. They propose a genetic algorithm that finds the set of all non-dominated solutions. Their criteria are makespan and consumption of a particular renewable resource. The initial population is formed generating precedence feasible solutions. The fitness function is evaluated using makespan values of the feasible solutions and penalizing the infeasible solutions with the excess amount of non-renewable resource consumption. Another fitness function proposed is the total consumption of renewable resources. A two-point crossover, one for carrying the chromosomes and the other for modes, is used. Two different mutation operators are used for diversification purposes. The parameter setting is done by using test instances in PSPLIB.

Pulat and Horn (1996) discuss the time-resource trade-off problem with linear time and resource costs. Resources, if there are several, are grouped into two. The resource consumption costs constitute two of the three objectives. The third objective is the makespan. Using these three objectives a set of efficient solutions is obtained. An important assumption of the problem is that the relationship between the resource cost and project duration is linear. With this assumption the problem can be formulated as a network problem with an activity on arc network and activities are crashed using the labeling procedure and flow augmenting paths. The resource cost objectives have weights associated with them and for each value of the weights the problem is solved and a set of all efficient solutions is generated. To

specify the weights and determine the efficient solutions for given weights enumerative and interactive approaches are proposed.

CHAPTER 4

OUR PROBLEM, ITS MODEL AND OPTIMALITY PROPERTIES

In this chapter we first define our problem together with its underlying assumptions. We then give the mathematical model. Finally we introduce our elimination rules that can be used to reduce the complexity of the solutions.

4.1 Problem Definition

We consider N non dummy activities. Activity 0 stands for a dummy starting activity and $N+1$ stands for a dummy ending activity. Activity i $(i=1,\ldots,N)$ has m_i modes.

Mode j $(j=1,\ldots,m_i)$ of activity i is characterized by the following parameters.

r_{ij}: units of resource required by activity i when executed in mode j

p_{ij}: duration (processing requirement) of activity i when executed in mode j

We assume all modes are efficient in the sense that $r_{ij} < r_{ik}$ implies $p_{ij} > p_{ik}$. In our convention, we assume that $p_{ij} < p_{ij+1}$ for all i, i.e., the modes of all activities are ordered in their nondecreasing order of processing times, hence nonincreasing order of resource requirement. Accordingly the first mode of each activity is its shortest duration mode and it consumes highest resource.

We use terms task and activity interchangeably throughout the thesis.

We let *T* be the length of the critical path. It is found by *CPM* after setting all activities to their shortest duration mode. *T* is the deadline of the project.

Activity *i* can start at any time *t* between 0 and *T-1*.

The precedence network is defined by the immediate predecessor sets. E_i is the set of immediate predecessors of activity *i.*

We assume that the single renewable resource is available at unlimited quantity. Moreover we make the following assumptions:

The tasks are nonpreemptive, i.e., once a task starts it should be processed till its completion.

The dummy activity *N+ 1* can start after all activities are complete. As *T* is the deadline of the project, the activity starts and completes at time *T*.

All parameters are integers, are known with certainty, i.e., the system is deterministic.

The parameters are not subject to any change, i.e., the system is static.

4.2 Mathematical Model

Our main decision variable x_{ijt} gives the start time and selected mode of each activity and is defined as

$$x_{ijt}: \begin{cases} 1, & \text{if activity } i \text{ starts at time } t \text{ and is executed with mode } j \\ 0, & \text{otherwise} \end{cases}$$

The decision variable *R* is defined as the amount of renewable resource available for each time unit.

Our constraints are explained next.

Each activity should be assigned to exactly one mode and starting time

$$\sum_{j\in m_i}\sum_{t=ES_i}^{LS_i} x_{ijt} = 1 \qquad \forall i \in A \qquad (1)$$

The resource consumption at any period cannot exceed the available resource

$$\sum_{i\in A}\sum_{j\in m_i} r_{ij} * \sum_{t=t}^{t=t+p_{ij}-1} x_{ijt} \le R \qquad \forall t \qquad (2)$$

The ending dummy activity must start at *T*.

$$x_{N+2,1,T} = 1 \qquad (3)$$

An activity can start only after all of its predecessors are completed.

$$\sum_{j\in m_i}\sum_{t=ES_i}^{LS_i} t * x_{ijt} \ge \sum_{j\in m_k}\sum_{t=ES_k}^{LS_k} (t + p_{kj}) * x_{kjt} \qquad i = 2,3,...,N+2; \quad \forall k \in E_i \qquad (4)$$

x_{ijt}'s are binary variables.

$$x_{ijt} = \ 0,1 \qquad i = 1,2,....,N+2 \text{ ; j=1,2,....,m}_i \text{ ; t=1,2,....T} \qquad (5)$$

The objective is to minimize the maximum renewable resource usage and is expressed as:

Minimize *R* (6)

The model contains $T * \sum_{i=1}^{N+2} m_i$ binary variables *(x_{ijt}s)* and one continuous variable *(R).*

There are $N + T + \sum_{i=1}^{N+2} E_i$ constraints.

4.3 Properties of the Optimal Solution

In this section we present some properties that are used to reduce the problem size. The problem size is reduced when any mode is eliminated and any start time is set. Properties 1, 4 and 5 are used to eliminate the modes that cannot lead to feasible or unique optimal solutions. Property 2 and Property 3 are used to set both the modes

and the starting times of the activities that cannot lead to unique optimal solutions. We use the following notation to state our properties.

Early Start Schedule

ES_i = Earliest start time of activity i, when all activities are at their shortest modes

EC_i = Earliest completion time of activity i, when all activities are at their shortest modes

Late Start Schedule

LS_i = Latest start time of activity i, when all activities are at their shortest modes

LC_i = Latest completion time of activity i, when all activities are at their shortest modes

LB = A lower bound on the objective function value for the problem

UB = An upper bound on the objective function value for the problem

We call an activity critical if its early start time is equal to its late start time. i.e., activity i is critical if and only if $ES_i = LS_i$. We set the critical activities' start times to their early start times and modes to their shortest duration modes. We further reduce the problem size by using the results of Property 2 and Property 3.

Property 1.

If $LC_i - ES_i < p_{ij}$, then for activity i modes $j, j+1, ..., m_i$ cannot lead to a feasible solution.

Proof: $LC_i - ES_i$ is the maximum processing time allowed for activity i to maintain feasibility. Hence, a solution in which activity i is assigned mode j such that $p_{ij} > LC_i - ES_i$, can never lead to a feasible solution. □

The properties 2 and 3 use the following additional notation.

L_{tE} = Resource usage at period t when all activities start at their earliest start time and shortest mode

L_{tL} = Resource usage at period t when all activities start at their latest start time and shortest mode

Property 2.

There exists an optimal schedule in which activity i is processed between ES_i and EC_i at its shortest duration mode if $L_{tE} \leq LB$ at every $t \in [0, EC_i]$

Proof: Assume a schedule A_1 that contradicts with the condition of the theorem. Activity i completes at time $C_i > EC_i$ at a longer mode and $L_{tE} \leq LB$ at every $t \in [0, EC_i]$. Shift activity i such that it completes at time EC_i at its shorter mode, and get schedule A_2 which is in line with the condition of the property.

Divide the time span into time intervals as $[0, EC_i]$ and $(EC_i, T]$ and let z_{ij} be the maximum resource usage of solution A_i in interval j. Figure 4.1 illustrates this situation.

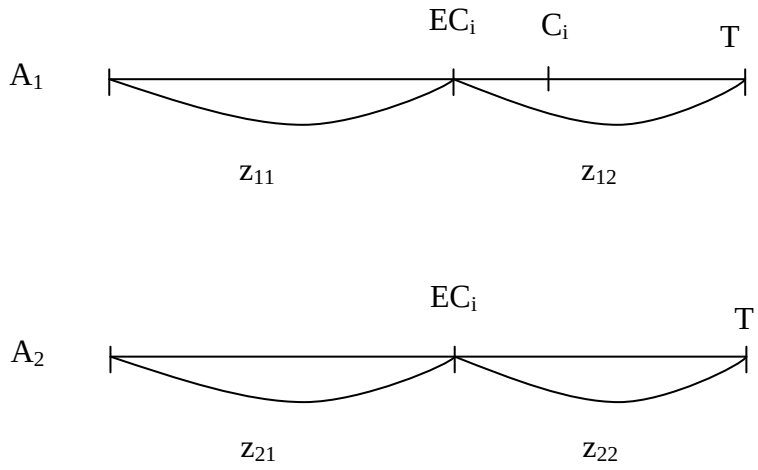

Figure 4.1 - A schedule that illustrates Property 2

In Figure 4.1, note that $z_{12} \geq z_{22}$ as in addition to all activities being processed in the interval $(EC_i, T]$, some portion of activity i is also processed for A_1. Also note that,

$z_{11} \le z_{21}$ as the resource consumed by activity i is less (as a smaller time consuming mode is used) for A_1 in the interval $[0, EC_i]$.

Let z_{A_i} be the maximum resource used by A_i. Note that $z_{A_i} \ge LB$

$z_{A_1} = Max\ \ z_{11}, z_{12}\ \ = z_{12}$ as $z_{11} \le z_{21} \le LB$ and $z_{A_1} \ge LB$ implies $z_{12} \ge LB$

$z_{A_2} = Max\ \ z_{21}, z_{22}\ \ = z_{22}$ as $z_{21} = L_{tE} \le LB$ and $z_{A_2} \ge LB$ implies $z_{22} \ge LB$

This follows $z_{A_1} \ge z_{A_2}$ as $z_{12} \ge z_{22}$. Hence a schedule that contradicts with the condition of the property can never be better. □

Property 3.

There exists an optimal schedule in which activity i is processed between LS_i and LC_i at its shortest duration mode if $L_{tL} \le LB$ at every $t \in [LS_i, T]$

Proof: Assume a schedule B_1 that contradicts with the condition of the theorem. Activity i starts at time $S_i < LS_i$ at longer mode and $L_{tL} \le LB$ at every $t \in [LS_i, T]$. Shift activity i such that it starts at time LS_i at its shorter mode, and get schedule B_2 which is in line with the condition of the property.

Divide the time span into time intervals as $[0, LS_i)$ and $[LS_i, T]$ and let z_{ij} be the maximum resource usage of solution B_i in interval j. Figure 4.2. illustrates this situation.

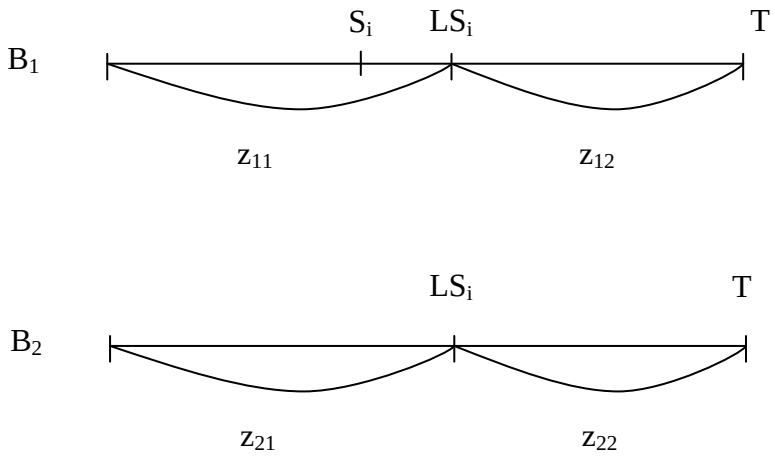

Figure 4.2 – A Schedule that illustrates Property 3

In Figure 4.2 note that $z_{11} \geq z_{21}$ as in addition to all activities being processed in the interval $(EC_i, T]$, some portion of activity i is also processed for A_1. Also note that, $z_{12} \leq z_{22}$ as the resource consumed by activity i is less (as a smaller time consuming mode is used) for B_1 in the interval $[LS_i, T]$.

Let z_{B_i} be the maximum resource used by A_i. Note that $z_{B_i} \geq LB$

$z_{B_1} = Max\ \ z_{11}, z_{12}\ \ = z_{11}$ as $z_{12} \leq z_{22} \leq LB$ and $z_{B_1} \geq LB$ implies $z_{11} \geq LB$

$z_{B_2} = Max\ \ z_{21}, z_{22}\ \ = z_{21}$ as $z_{22} = L_{tE} \leq LB$ and $z_{B_2} \geq LB$ implies $z_{21} = z_{B_2}$

This follows $z_{B_1} = z_{11} \geq z_{21} = z_{B_2}$. Hence a schedule that contradicts with the condition of the property can never be better. □

For the reduced problem with non-fixed start times and modes, we introduce two more properties to eliminate the modes. The eliminated modes are the ones that either lead to non promising or infeasible solutions.

We define the following sets that are used to state our mode elimination properties.

S_t : {Set of activities that have to be processed at time t}

Mathematically,

$S_t = \{ i \mid (EC_i - LS_i > 0) \wedge (t \in [LS_i, EC_i] \}$

FR_i : {Set of activities that are processed in parallel with activity i for at least one time unit}

Mathematically, $FR_i = \{k \mid i \in S_t \wedge k \in S_t, \forall t\}$

FS_i : {Maximal set for the activities that can be processed together with activity i }

Mathematically, $FS_i = \{k \mid (LC_i > ES_k) \wedge (ES_i < LC_k) \wedge (k \notin Succ_i) \wedge (k \notin E_i)\}$

Property 4.

If $r_{ik} + \sum_{j \in FR_i} r_{jm_j} > UB$, then for activity i modes 1, …, k cannot lead to an optimal solution.

Proof: In at least one time unit, the activities in FR_i should be processed together with activity i. $\sum_{j \in FR_i} r_{j1}$ is a lower bound on the total resource consumption of the activities that should be processed with activity i. So, when activity i is assigned to mode k, and all the activities in FR_i are assigned to their minimum resource consuming modes, the maximum resource usage is no smaller than $r_{ik} + \sum_{j \in FR_i} r_{jm_j}$.

This follows, a solution in which activity i is assigned to mode k is dominated by the upper bound (UB) solution. □

Property 5.

Recall that FS_i is the maximal set for the activities that can be processed together with activity i. If $r_{ik} + \sum_{j \in FS_i} r_{j1} \leq LB$, then there exists an optimal schedule in which activity i is assigned to modes $1, ..., k$.

Proof: $r_{ik} + \sum_{j \in FS_i} r_{jm1}$ is an upper bound on the resource usage for the time points at which activity i is processed at mode k. If this upper bound is no bigger than LB, then setting activity i to its lower resource usage modes will never improve the objective function value that is surely greater than LB. □

4.4 Illustration of Properties on the Example Problem

We illustrate the properties on our example problem. In the examples we do not consider the critical activities as they are already set to their first modes to meet the minimum project completion time. Table 4-1 shows the CPM calculations for the example problem and Table 4-2 shows the mode information.

Table 4-1 – The *CPM* calculations for the example problem

Activity	**Immediate Predecessors**	**Duration (days)**	ES_i	EC_i	LS_i	LC_i	$Slack_i$
1 (dummy)	-	-	0	0	0	0	0
2	1	7	0	7	3	10	3
3	1	9	0	9	7	16	7
4	1	8	0	8	0	8	0
5	2	8	7	15	11	19	4
6	4	6	8	14	10	16	2
7	4	8	8	16	8	16	0
8	2,4	9	8	17	10	19	2
9	4	7	8	15	9	16	1
10	5,8	6	17	23	19	25	2
11	3,6,7,9	9	16	25	16	25	0
12 (Dummy)	10,11	-	25	25	25	25	0

Table 4-2– The execution modes for non-dummy activities

ACTIVITIES		MODES																			
	i	p_{i1}	r_{i1}	p_{i2}	r_{i2}	p_{i3}	r_{i3}	p_{i4}	r_{i4}	p_{i5}	r_{i5}	p_{i6}	r_{i6}	p_{i7}	r_{i7}	p_{i8}	r_{i8}	p_{i9}	r_{i9}	p_{i10}	r_{i10}
	2	7	8	9	6	11	5	13	4	18	3	27	2	55	1	-	-	-	-	-	-
	3	9	11	10	9	11	8	13	7	15	6	19	5	23	4	31	3	47	2	95	1
	4	8	8	9	7	10	6	12	5	16	4	21	3	32	2	64	1	-	-	-	-
	5	8	10	9	8	10	7	12	6	14	5	18	4	24	3	37	2	74	1	-	-
	6	6	8	7	6	9	5	11	4	15	3	22	2	45	1	-	-	-	-	-	-
	7	8	9	9	7	11	6	13	5	16	4	22	3	33	2	66	1	-	-	-	-
	8	9	11	10	9	12	8	13	7	16	6	19	5	24	4	32	3	48	2	97	1
	9	7	9	8	7	9	6	11	5	14	4	19	3	29	2	59	1	-	-	-	-
	10	6	8	7	6	8	5	10	4	14	3	21	2	43	1	-	-	-	-	-	-
	11	9	9	10	8	11	7	13	6	16	5	20	4	27	3	40	2	81	1	-	-

Property 1.

Consider a noncritical activity, say activity 5. The activity cannot start before its earliest start, 7, due to its predecessors and cannot be completed later than its latest completion time due to its successors and the deadline. The total amount of time allowed for activity 5 is $LC_5 - ES_5 = 19 - 7 = 12$. We eliminate all modes of the activity having processing requirements of more than 12 time units. Hence, modes 5 through 9, are eliminated from further considerations as they cannot lead to feasible solutions. This leaves 4 modes for activity 5.

Property 2.

Assume we have a lower bound LB=30 (in the following sections it will be shown that 30 is a valid lower bound for the problem). The resource profile of the early start schedule is given in Figure 4.3. The horizontal line in the figure shows the lower bound value.

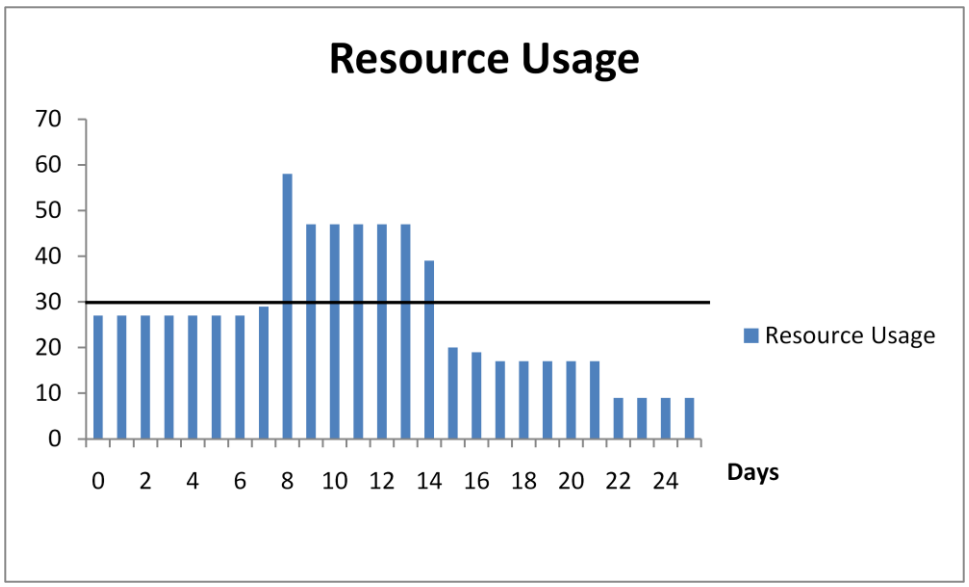

Figure 4.3 – The resource usage profile

As can be seen in Figure 4.3 the resource usage first exceeds the lower bound in period 8, i.e., $L_{t_E} = 8$. So according to the property if any activity using at its shortest duration mode (maximum resource consuming mode) can be completed before L_{t_E} in the early start schedule, there exists an optimal solution in which that activity is scheduled to start at its earliest start using its shortest duration mode. The earliest completion time of noncritical activity 2 is 7 (see Table 4-1), which is smaller than $L_{t_E} = 8$. So in at least one optimal solution, activity 2 starts at its earliest start time at its shortest duration mode.

Property 3 is not illustrated as it uses the same idea with Property 2. It uses late start schedules in place of early start schedules and sets the optimal start times to their latest start times in place of early start times.

Property 4.

Assume we have an upper bound UB=37. We select activity 5 and form set FR_i. Activity 5 has to be processed between periods 11 through 15, i.e., $activity\ 5 \in S_{11}, S_{12}, S_{13}, S_{14}$ and S_{15}. Among the five sets, S_{11} is the one that contains

highest number of activities. The activities in S_{11} are 5, 6, 7, 8 and 9 and $FR_5 = \{6,7,8,9\}$. Now we check the condition of property 4 and compare the minimum resource consumption in Set FR_5 with UB. Note that $\sum_{j \in FR_5} r_{jm_j} = 1+1+1+1 = 4$. As $4 + r_{5j} < 37$ for all r_{5j} values (all are no bigger than 10), no mode of activity 5 could be eliminated.

Our data set includes instances with small values of resource consumptions (no bigger than 10), hence we could not benefit from Property 4.

Property 5.

Assume we have a lower bound LB=30. A noncritical activity 10 has the smallest $[ES_i, LC_i]$ interval length. The activities that have a chance to be processed with activity 10 are 5, 8 and 11, hence they constitute set FS_i. Now we check the condition of property 5 and compare the resource consumption in Set FS_{10} with LB. $\sum_{j \in FS_i} r_{j1} = 9$ and the maximum resource consuming mode of activity consumes 8 units of resource. $8 + 9 = 17 < 30 = LB$, so any mode consuming less resource is eliminated as using that mode cannot change the optimal.

CHAPTER 5

SOLUTION APPROACHES

The time/resource trade-off problem defines the completion time of each activity together with its mode. Our time/resource trade-off problem aims to minimize the maximum consumption of a single renewable resource. Möhring (1984) shows that the resource investment problem with single renewable resource is strongly NP-hard. So is our problem, with an additional complexity brought by mode decisions.

In this chapter we present our present our solution approaches. Section 5.1 and Section 5.2 present our lower bounding procedures and heuristic procedure, respectively and Section 5.3 illustrates these on the example problem.

5.1 Lower Bounding Procedures

We develop four lower bounds each of which provides an underestimate of the objective function value. We use those estimates to evaluate the performance of our heuristic procedure.

Lower Bound 1:

The first lower bound is based on the overlapping activities. We call activity i and activity k as overlapping if they should be processed together in at least one time unit. We let a_{ik} denote the length of the biggest interval in which both activity i and activity k must be processed.

For all pairs of activities, $ES_i \le ES_k$, we calculate a_{ik} values. If their minimum possible total processing times (when set to their first modes) is smaller than a_{ik}, activities i and k will be processed in parallel for at least one time unit. We let FR denote the set of such activity pairs.

$$FR:\ (i,k)\,|\,a_{ik} < p_{i1} + p_{k1} \quad \text{where } a_{ik} = Max\{LC_i, LC_k\} - Min\{ES_i, ES_k\}$$

We find a lower bound considering the activities in set FR and their minimum resource consuming modes. For each activity pair in the set, the minimum possible consumption is calculated and their maximum is selected for the lower bound. The lower bound LB_1 becomes

$$LB_1 = \underset{(i,k)\in FR}{Max}\ r_{im_i} + r_{km_k}$$

If $FR = \varnothing$, then an extended set, FRE is defined. Set FRE includes the activity pairs that should be performed in parallel for at least one time unit at their maximum duration modes. Formally,

$$FRE:\ (i,k)\,|\,a_{ik} < p_{im_i} + p_{km_k}$$

Using Set FRE, we calculate a lower bounding procedure whose algorithmic description is provided in the next page.

Algorithmic explanation of lower bounding procedure which uses set *FRE*

For all $(i,k) \in FRE$;

Let a and b be the modes of activities i and k respectively;

For all a starting from $a = m_i$ and decreasing to $a = 1$

{ For all b starting from $b = m_k$ and decreasing to $b = 1$

$if\ (t_{ia} + t_{kb}) \le a_{ik}$

$res1 = Max\{r_{ia}, r_{kb}\}$

End for

}

For all b starting from $b = m_k$ and decreasing to $b = 1$

{ For all a starting from $a = m_i$ and decreasing to $a = 1$

$if\ (t_{ia} + t_{kb}) \le a_{ik}$

$res1 = Max\{r_{ia}, r_{kb}\}$

End for

}

$resm(i,k) = Min\{res1, res2\}$

$LB_1 = \underset{(i,k) \in FRE}{Max} \{resm(i,k)\}$

Lower Bound 2:

If the earliest completion time of an activity is later than its latest start time, then the activity has to be processed between its latest start and earliest completion times. Formally, for activity i if $EC_i - LS_i \geq 0$ then it is certainly processed in interval $[LS_i, EC_i]$. If there is an interval in which the activity is definitely processed, then for each time t, the set of activities that have to be processed can be defined as well. So, we let S_t denote the set of activities that have to be processed at time t. Then, for all the time points, the activities in the set S_t are found and their minimum possible resource consumptions are summed up. The minimum resource consumption is found by setting all activities to their minimum resource usage mode. The maximum of the minimum consumptions overall periods constitutes the lower bound.

So, Mathematically,

if $EC_i - LS_i \geq 0$, $i \in S_t \quad \forall t \in [LS_i, EC_i]$

$$LB_2 = \underset{t}{Max} \left\{ \sum_{i \in S_t} r_{im_i} \right\}$$

Lower Bound 3:

The workload of an activity is the total resource it consumes during its execution. The workload of activity i (WL_i), at its selected mode j, is $p_{ij} * r_{ij}$, and the minimum workload (MWL_i) for activity i over all its modes is

$$MWL_i = \underset{j}{Min} \; p_{ij} * r_{ij}$$

An activity i, should be completed before its latest completion time, LC_i. It means that, it consumes at least MWL_i amount of resource before LC_i. The minimum possible total resource consumed by the end of LC_i, $MTWL_i$, is the sum of

minimum workloads of activities whose latest completion is no later than LC_i. Formally,

$MTWL_i = \sum_{k \in Fin_i} MWL_k$ where

$Fin_i : \ k \,|\, LC_k \le LC_i$

To get a lower bound on the maximum resource consumption, the total workload is distributed evenly in the interval $[0, LC_i]$ and the resulting value is rounded up to the smallest integer value.

$$Min_i = \left\lceil \frac{MTWL_i}{LC_i} \right\rceil$$

To find a lower bound, all activities are considered and maximum of Min_i values is selected. So, a valid lower bound is $\underset{i}{Max} \ Min_i$.

Note that this bound considers all activities that should complete no later than LC_i to find Min_i. However there may exist an activity such that $LC_k > LC_i$, but $LS_k < LC_j$. For such an activity, the minimum processing before LC_i is $LC_i - LS_k$. We include this portion by weighing with the minimum weight r_{km_k}, hence obtain a lower bound on the total workload of activity k.

The updated $MTWL_i$ values become $\sum_{k \in Fin_i} MWL_k + \sum_{k \in St_i} (LC_i - LS_k) * r_{km_k}$ where

$St_i : \ k \,|\, LC_k > LC_i \wedge LS_k \le LC_i$

The following expression, LB_3 gives a valid lower bound;

$$LB_3 = \left\lceil \underset{i}{Max} \left\{ \sum_{k \in Fin_i} MWL_k + \sum_{k \in St_i} (LC_i - LS_k) * r_{km_k} \right\} \right\rceil$$

Lower Bound 4: Linear Programming (LP) Relaxation with Valid Cuts

We simply relax the integrality constraints on x_{ijt} variables and get the following relation.

$$0 \le x_{ijt} \le 1$$

The resulting model is the *LP* relaxation of the original model. The optimal solution to the *LP* relaxation provides a lower bound for our minimization problem.

In order to strengthen the relaxation we include some valid cuts (constraints). These cuts are the relations that are satisfied by the integer problem however not by the associated linear program.

Cut 1: Reduction due To Feasibility

The first cut is the one proposed by Akkan et al. (2005) to the discrete time –cost trade-off problem. The cut is valid for our problem and is explained below.

Consider any pair of activities (i,k) such that $k \in E_i$ and any combination of modes $a \in M_i$ and $b \in M_k$

If $LC_i - ES_k < p_{ia} + p_{kb}$ then modes, $M_{i+1},....,M_{m_i}$ and $M_k, M_{k+1},....,M_{m_k}$ cannot provide a feasible solution. So, a valid cut is as follows;

$$\sum_{t=1}^{T}\sum_{j=a}^{m_i} x_{ijt} + \sum_{t=1}^{T}\sum_{j=b}^{m_k} x_{kjt} \le 1 \quad \forall (i,k) \in FC \text{ ; where}$$

$$FC = \{(i,k) \,|\, k \in E_i \wedge (LC_i - ES_k < p_{ia} + p_{kb})\}$$

Cut 2: Reduction due to Optimality

Let activity *i* and activity *k* be two activities that have to be processed together for at least one time period, i.e., (i, k) is in Set FR. If their resource usage exceeds a given

upper bound (UB) for any mode combinations $a \in M_i$ and $b \in M_k$ then performing these activities in modes $M_1,....,M_i$ cannot lead to an optimal solution. So, the second cut can be stated as follows;

$$\sum_{t=1}^{T}\sum_{j=1}^{a} x_{ijt} + \sum_{t=1}^{T}\sum_{j=1}^{b} x_{kjt} \le 1 \qquad \forall (i,k) \in OC \text{ where}$$

$$OC = \{(i,k) \,|\, k \in FR \wedge (r_{ia} + r_{kb} \ge UB)\}$$

Together with these two cuts, our strengthened *LP* relaxation model is stated below. For the sake of completeness we give the previously stated constraints as well.

Min R (1)

s.to.

$$\sum_{j \in m_i}\sum_{t=ES_i}^{LS_i} x_{ijt} = 1 \qquad \forall i \in A \qquad (2)$$

$$\sum_{i \in A}\sum_{j \in m_i} r_{ij} * \sum_{t=t}^{t=t+p_{ij}-1} x_{ijt} \le R \qquad \forall t \qquad (3)$$

$$x_{N+2,1,T} = 1 \qquad (4)$$

$$\sum_{j \in m_i}\sum_{t=ES_i}^{LS_i} t * x_{ijt} \ge \sum_{j \in m_k}\sum_{t=ES_k}^{LS_k} (t + p_{kj}) * x_{kjt} \qquad i = 2,3,...,N+2; \;\; \forall k \in E_i \qquad (5)$$

$$\sum_{t=1}^{T}\sum_{j=a}^{m_i} x_{ijt} + \sum_{t=1}^{T}\sum_{j=b}^{m_k} x_{kjt} \le 1 \qquad \forall (i,k) \in FC \qquad (6)$$

$$\sum_{t=1}^{T}\sum_{j=1}^{a} x_{ijt} + \sum_{t=1}^{T}\sum_{j=1}^{b} x_{kjt} \le 1 \qquad \forall (i,k) \in OC \qquad (7)$$

$$x_{ijt} \ge 0 \qquad (8)$$

We round the optimal objective function value to the smallest following integer, to get a lower bound (LB_4) on the optimal objective value. Formally,

$LB_4 = \lceil z^*_{LP} \rceil$, where z^*_{LP} is the objective function value of the optimal LP Relaxation.

The overall lower bound is $LB = \underset{i=1,2,3,4}{Max}\{LB_i\}$. Note that LB is the best of our four lower bounds.

We include additional cuts using the results of Property 2 and Property 3. We let;

ESO: {Set of activities that are set to their first mode and start on their early start times according to Property 2}

LSO: {Set of activities that are set to their first mode and start on their late start times according to Property 3}

After these sets are specified, we add the following constraint sets to the linear programming relaxation of the problem.

$$x_{i1ES_i} = 1 \qquad \forall i \in ESO$$

$$x_{i1ES_i} = 1 \qquad \forall i \in LSO$$

In defining ESO and LSO we first use $LB = \underset{i=1,2,3}{Max}\ LB_i$

After we solve the LP Relaxation, we redefine ESO and LSO using $LB = \underset{i=1,2,3,4}{Max}\ LB_i$. With new ESO and LSO sets we resolve the LP Relaxation and get new LB_4, and update LB.

5.2 Heuristic Procedure

In this section we present our heuristic procedure that aims to find high quality feasible solutions to our problem. We evaluate the performance of our heuristic procedure relative to the optimal solutions for the problems for which optimal solution is available. For large sized problem instances for which the optimal solutions are not available, we make the evaluation relative to the best lower bound found in Section 5.1.

Our heuristic procedure runs in two steps: Construction and Improvement. The construction step finds initial feasible solutions. The improvement step improves the solutions that are found in construction step. Below is the detailed description of each step.

Step 1. Construction

The construction step proceeds in two steps:

Step 1.1. Finding mode assignments

Step 1.2. Finding the start times, given the mode assignments of step 1.1.

Step 1.1 finds the mode assignments by using the optimal solution of the LP relaxation. We retain the integer assignments and move the fractional modes to their next smaller duration modes. Our aim is to find a feasible solution while increasing the resource consumption as small as possible.

Formally we set activity i to mode k if $p_{ik} < p_{iLPR} < p_{i,k+1}$, hence $r_{ik} > r_{iLPR} > r_{i,k+1}$ where

p_{iLPR} = the duration of activity i in the LP relaxation solution.

r_{iLPR} = the resource consumption of activity i in the LP relaxation solution.

Note that by setting activity i to mode k, we guarantee feasibility. The resource consumption may increase slightly, as the new mode assignments require more resource.

Given the mode assignments, Step 1.2. defines the starting times of the activities, hence forms feasible schedules. The schedules are formed in three different ways each of which is discussed below.

Schedule 1. Early Start Schedule

Using the mode assignments found in Step 1.1. and applying *CPM*, we set the start time of activity i to ES_i for Schedule 1.

Schedule 2. Late Start Schedule

Using the mode assignments found in Step 1.1. and applying *CPM*, we set the start time of activity i to LS_i for Schedule 2.

Schedule 3. Alternating Early Start – Late Start (AEL) Schedule

Using the mode assignments found in Step 1.1. we calculate ES_i and LS_i. Then we sort activities according to their early start and late start times and put them into the following sets.

$Sorted_{ES}$ = Set of activities in nondecreasing order of their early start times

$Sorted_{LS}$ = Set of activities in nonincreasing order of their late start times

First the start time of the first not yet scheduled activity in $Sorted_{ES}$ is set to its early start time. Then the start time of the first not yet scheduled activity in $Sorted_{LS}$ is set to its late start time. The procedure continues until all activities are scheduled.

Step 2. Improvement

Each schedule found in Step 1 is subjected to an improvement procedure with the hope of reducing the maximum resource consumption.

Given the schedule, we consider the time point that defines the maximum resource consumption and try to reduce the load of this time point. The reduction can be done in three different ways.

- Reducing the duration of an activity
- Increasing the duration of an activity
- Changing the start time of an activity

When the duration is reduced, keeping the start time of the activity same, the activity will no longer be processed at some time points. Hence, the load on the maximum load time point may be removed and maximum resource consumption might decrease. But, the reverse is also possible. By reducing the duration of an activity, the resource consumption of that activity increases. This, in the case of still

being processed on maximum load time point, increases the maximum resource consumption.

When the duration of an activity is increased, the activity consumes less resource, which leads to a reduction in the resource usage in the maximum load time point. Again, the reverse is also possible. Increasing the time of an activity causes the activity spread its workload to more time points. So, a time point, in which the activity is not processed before but will be processed now, will have more resource consumption than it had before the duration change.

When the start time of an activity is increased or decreased, the maximum load may shift to another time point, which in turn might reduce (increase) the maximum resource consumption.

Below is the detailed algorithmic description of our improvement procedure.

The main steps of the Heuristic Algorithm

Heuristic Algorithm

Using each construction heuristic as an initial solution;

Execute OnlyImprove

Execute BestAvailable

Choose the best solution reached.

The OnlyImprove step is the main body of the algorithm. The step calls ModeChange module to change the modes of the activities and SlideAct module to change the start times of the activities.

The OnlyImprove Module

OnlyImprove

Repeat

Increase iter by 1

Execute ModeChange by counter1 times

If maximum resource usage is less

Change current solution

counter1=1; counter2=1;

Else

Return to the solution before ModeChange

counter1 increases by 1

Execute SlideAct by counter2 times

If maximum resource usage is less

Change current solution

counter1=1; counter2=1;

Else

Return to the best solution available for the given initial solution

counter2 increases by 1

Until iter reaches iterlim or one of the counters reaches nonimplim.

In our implementation, we set iterlim to 100 and nonimplim to 7.

The BestAvailable Module

BestAvailable

Repeat

Increase niter by 1;

Execute ModeChange once;

If maximum resource usage is less

counter1=1; counter2=1;

Execute OnlyImprove with nonimplim=5.

Else

counter1 increases by 1.

Change current solution.

Execute SlideAct once;

If maximum resource usage is less

counter1=1; counter2=1;

Execute OnlyImprove with nonimplim=5.

Else

counter2 increases by 1.

Change current solution.

Until niter reaches niterlim or one of the counters reaches nonimprolim.

In our implementation, we set niterlim to 1000 and nonimprolim to 150.

The ModeChange Module

maxt: the first time when the resource usage is at its highest level.

S_{maxt}: The set of activities executed at time maxt

ModeChange (changing the mode of an activity)

For $i \in S_{maxt}$

Change the mode assignment of activity i with the next smaller mode (if possible) with all other mode assignments remaining the same.

Adjust all successor and predecessor activities' early start, late start and start times.

Calculate the maximum resource usage.

Change the mode assignment of activity i with the next larger mode (if possible) with all other mode assignments remaining the same.

Adjust all successor and predecessor activities' early start, late start and start times.

Calculate the maximum resource usage.

Choose the assignment with the least maximum resource usage - Return the transformed schedule and its maximum resource usage.

The SlideAct Module

SlideAct (changing the start time of an activity)

For $i \in S_{\max t}$

Slide activity i up to its earliest start;

Adjust all succeeding and preceding activities' starting times;

Calculate the maximum resource usage;

Slide activity i up to its latest start;

Adjust all succeeding and preceding activities' starting times.;

Calculate the maximum resource usage;

Choose the assignment possibility with the least maximum resource usage - Return the transformed schedule and its maximum resource usage.

Our search procedure resembles the Tabu Search algorithm. We visit neighboring solutions and let non-improving solutions only when an improved solution cannot be reached after a specified number of iterations. When we find an improved solution, we search without changing the current solution on hand as in intensification phase of tabu search algorithms. When there is no improvement for specified number of iterations, we switch to a new starting solution as in diversification phase of tabu search algorithms.

5.3 Illustration of Solution Procedures on an Example Problem

We illustrate the lower bounds and heuristic procedures on our example problem. In Table 5.1 we restate the CPM calculations and mode data, but now with not-yet-eliminated modes. These modes are shown in bold faces and larger font size in Table 5-2.

Table 5-1- The *CPM* calculations for the example problem

Activity	**Immediate Predecessors**	**Duration (days)**	ES_i	EC_i	LS_i	LC_i	$Slack_i$
1 (dummy)	-	-	0	0	0	0	0
2	1	7	0	7	3	10	3
3	1	9	0	9	7	16	7
4	1	8	0	8	0	8	0
5	2	8	7	15	11	19	4
6	4	6	8	14	10	16	2
7	4	8	8	16	8	16	0
8	2,4	9	8	17	10	19	2
9	4	7	8	15	9	16	1
10	5,8	6	17	23	19	25	2
11	3,6,7,9	9	16	25	16	25	0
12 (Dummy)	10,11	-	25	25	25	25	0

Table 5-2– The execution modes for non-dummy activities

ACTIVITIES		MODES																			
	i	p_{i1}	r_{i1}	p_{i2}	r_{i2}	p_{i3}	r_{i3}	p_{i4}	r_{i4}	p_{i5}	r_{i5}	p_{i6}	r_{i6}	p_{i7}	r_{i7}	p_{i8}	r_{i8}	p_{i9}	r_{i9}	p_{i10}	r_{i10}
	2	7	8	9	6	11	5	13	4	18	3	27	2	55	1	-	-	-	-	-	-
	3	9	11	10	9	11	8	13	7	15	6	19	5	23	4	31	3	47	2	95	1
	4	8	8	9	7	10	6	12	5	16	4	21	3	32	2	64	1	-	-	-	-
	5	8	10	9	8	10	7	12	6	14	5	18	4	24	3	37	2	74	1	-	-
	6	6	8	7	6	9	5	11	4	15	3	22	2	45	1	-	-	-	-	-	-
	7	8	9	9	7	11	6	13	5	16	4	22	3	33	2	66	1	-	-	-	-
	8	9	11	10	9	12	8	13	7	16	6	19	5	24	4	32	3	48	2	97	1
	9	7	9	8	7	9	6	11	5	14	4	19	3	29	2	59	1	-	-	-	-
	10	6	8	7	6	8	5	10	4	14	3	21	2	43	1	-	-	-	-	-	-
	11	9	9	10	8	11	7	13	6	16	5	20	4	27	3	40	2	81	1	-	-

Lower Bound 1

Consider activities 5 and 9. The latest possible start time for both activities is 19, and the earliest possible start time for one of them is 7. This follows both must be completed in at most 12 time units, however their smallest duration modes add up to 15 time units. So, surely the resource consumed by the activities will add up to, at least, the sum of their minimum resource consumptions. That is 6+7=13. 13 is our first lower bound value.

Lower Bound 2

As in example 4.4.3 let us take time point 11. $S_{11} = \{5,6,7,8,9\}$. Note that these activities are in set S_{11} because their processing interval, $[LS_i, EC_i]$, includes 11. For example the interval for activity 7 is $[8,16]$. The minimum total resource consumption of these five activities over their non-eliminated modes equals to 37; mathematically;

$$\sum_{i \in S_{11}} r_{im_i} = r_{54} + r_{62} + r_{71} + r_{82} + r_{92} = 6 + 6 + 9 + 9 + 7 = 37$$

For the example problem, our second lower bound, by chance, equals the objective function value.

Lower Bound 3:

The lower bound on the resource consumption is calculated using the minimum workloads. Table 5-3 illustrates the minimum workloads of the activities sorted by their latest completion time.

According to the Table 5.3, the maximum average workload is observed when $t = 19$. We illustrate the lower bound only for the maximum workload period, $t = 19$. Some portion of activity 11 must be carried out before $t = 19$ as its latest start is 16 and minimum processing time is 9. The one third ($\frac{3}{9}$) of the activity must be processed before $t = 19$. As the minimum workload of 11 is 77, the per unit time increase in the workload is $\frac{1}{19} * \frac{77}{3} = 4.05$. Hence the lower bound improves to $27.32 + 4.05 = 31.37$. It can be seen that the lower bound has ameliorated by approximately 15% with the inclusion of the additional workload.

Table 5-3 – Lower Bound 3 calculation for the example problem

Activity #	**Latest Completion**	**Minimum Workload**	**Minimum Cumulative Workload**	**Minimum resource consumed**
4	8	52	52	52/8=6.50
2	10	88	52+88=140	140/10=14.00
3	16	60	375	23.44
6	16	70	375	23.44
7	16	42	375	23.44
9	16	63	375	23.44
5	19	90	519	**27.32**
8	19	54	519	**27.32**
10	25	40	559	22.36
11	25	77	636	25.44

Cuts for the *LP* Relaxation

To illustrate Cut 1, two activities that are related in the precedence network are considered. Note that activities 2 and 5 are related as $2 \in E_5$. We find that $LC_5 - ES_2 = 19$.

Now, consider the maximum resource consuming modes of activities 2 and 5, i.e., mode 1 of activity 2, and mode 4 of activity 5. The associated total resource consumption is 19, i.e., no other mode combinations of the activities can exceed 19. Therefore no cut can be generated considering these two activities.

However if no earlier mode eliminations had been done, mode 2 of activity 2 and mode 4 of activity 5 together would form a cut as their total resource consumption exceeds 19. The cut supporting this result is expressed below:

$$\sum_{t=1}^{25}\sum_{j=4}^{9} x_{5jt} + \sum_{t=1}^{25}\sum_{j=2}^{7} x_{2jt} \leq 1$$

Similarly mode 1 of activity 2 and mode 5 of activity 5 would form a cut as their total resource consumption exceeds 19. This would add the following cut to the LP problem.

$$\sum_{t=1}^{25}\sum_{j=5}^{9} x_{5jt} + \sum_{t=1}^{25}\sum_{j=1}^{7} x_{2jt} \leq 1$$

Unfortunately, we could not generate any relation to support our second cut.

ModeChange Procedure

In this procedure, we try to improve the solution by changing the mode of an activity being processed at the maximum resource usage (bottleneck) period. We first find the maximum resource usage period. Figure 5.1 gives the resource profile, i.e., a graph showing the resource consumptions over time.

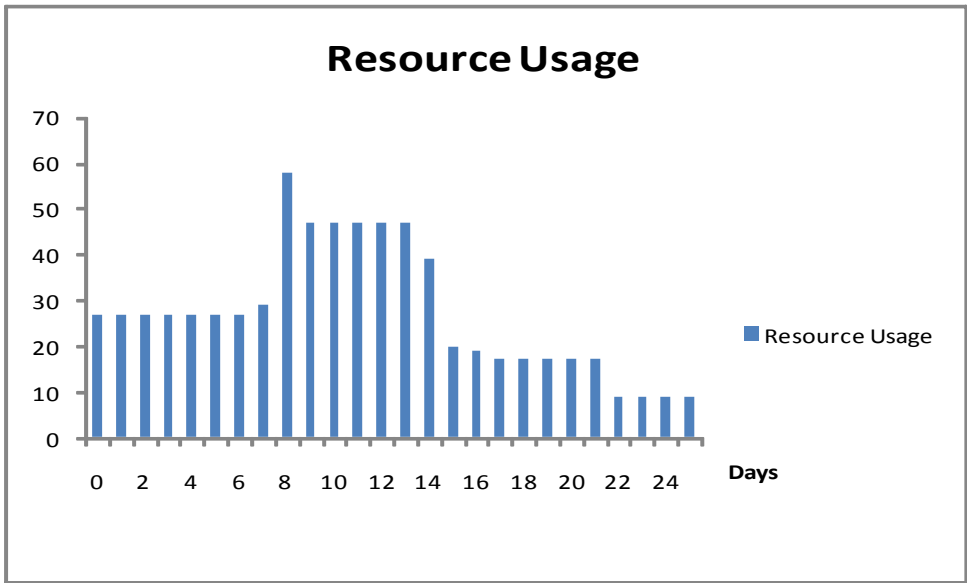

Figure 5.1 – Resource Profile for early start schedule

Figure 5.1 shows that the peak resource usage is at period 8 with resource consumption of 57 units. At that period, the noncritical activities in process are 3, 5, 6, 7, 8 and 9. All five activities indicated are subjected to the ModeChange procedure. As the activities are already at their first modes, their second modes are tried. The best improving solutions are found at mode 2 of activity 6 and mode 2 of activity 8, each returning two units of reduction in the maximum resource consumption. We break the tie in favor of activity 6 as it has lower index.

SlideAct Procedure

The procedure is again applied to the maximum resource consuming period. Let us consider the same initial schedule. The activities whose starting time can be changed are the ones with positive slack values. These activities are 3, 5, 6, 8 and 9. Among these activities, the best improvement is achieved by one unit sliding of activity 6 such that it starts at $t = 9$. By this change the maximum resource consuming period switches from period 8 to period 9 and the maximum load reduces from 57 to 54.

CHAPTER 6

COMPUTATIONAL EXPERIMENTS

In this chapter, our aim is to test the performance of our problem size reduction techniques, lower bounds and upper bounds. First, we discuss the generation of test problems, then we present our performance measures and lastly we discuss our results for the experiment.

6.1 Data Generation

Number of activities (*N*):

The problem is solved with 6 different problem sizes. The number of activities is selected as 10, 20, 30, 40, 60 and 80. For each problem size 10 instances are solved, so a total number of 60 instances are used to test the algorithm. We used a two phase methodology to generate these instances. In the first phase we generate the project network using Project Scheduling Instance Generator *(ProGen)* of Kolisch and Sprecher (1996). In the second phase we generate random processing times for activities and using these times, we find activity modes. *ProGen* is not used for mode generation as it fails to generate more than 5 modes for an activity.

We next explain our problem parameters;

Project Network:

For the Activity on Node *(AoN)* representation, coefficient of network complexity *(CNC)* is defined as the number of precedence relations per node. It means increasing *CNC* results in a more interconnected network. Alvarez-Valdes and Tamarit(1989) have shown that as the network gets more complex, the time required to solve the problem of those networks decreases. In our problem sets, networks of complexity 1.5 is used which is the lowest complexity preferred. In order to observe the effect of precedence network, we also include problem instances with higher *CNC* value of 2.1. For those networks we set $N = 20$ and $N = 80$ and generate 10 problem instances for each N . To see the effect of number of modes, we use 20 instances, 10 with $N = 20$ and 10 with $N = 80$. The instances are generated by decreasing the number of modes to half of its original setting. In doing so we take the odd modes, and skip the even ones. As a total our problem set includes 100 problem instances.

Processing Times and Resource Requirements:

We adapt the random generation scheme of Ranjbar and Kianfar (2007) to generate our parameters. In order to generate the modes systematically, we first generate processing times of the activities. The processing time of each activity is generated from a uniform probability distribution between 10 and 100. Then the following procedure is applied; $p_{i1} = \left\lfloor \sqrt{w_i} \right\rfloor$; $r_{i1} = \left\lceil \frac{w_i}{p_{i1}} \right\rceil$

$r_{ij+1} = r_{ij} - 1$; $p_{ij+1} = \left\lfloor \frac{w_i}{r_{ij+1}} \right\rfloor$, if efficient $(p_{ij+1} > p_{ij})$ accept; otherwise

$r_{ij+1} = r_{ij} - 2$; $p_{ij+1} = \left\lfloor \frac{w_i}{r_{ij+1}} \right\rfloor$, if efficient $(p_{ij+1} > p_{ij})$ accept; otherwise

…

Generate modes until $r_{ij} = 1$.

The processing time of the activities define the number of modes. The maximum number of modes is 10 as $p_{i1} = \left\lfloor \sqrt{100} \right\rfloor = 10$ and $r_{i1} = \left\lceil \frac{100}{10} \right\rceil = 10$. By decreasing r_{i1} with unit increments, a maximum of 10 modes can be found. Similarly, minimum number of modes is 3.

6.2 Performance Measures

To evaluate the performance of our problem size reduction techniques, lower bounds and upper bounds, we define some performance measures. In this section, we describe our performance measures.

The performance measures for problem size reduction techniques are

- Number of activities for which mode and time decisions wait to be taken
- Number of activity modes reduced by reduction properties
- Percent Reduction in total number of modes

The performance measures for lower bounds are

- Percent deviation from the optimal objective function value (for the problems solved to optimality)
- CPU Time (in seconds)
- Frequency of defining the best lower bound
- Frequency at which optimal = lower bound

The performance measures heuristics are

- Percent deviation from the optimal objective function value (For the problems solved to optimality)
- Frequency at which optimal = heuristic solution (For the problems solved to optimality)
- Percent deviation from the best lower bound (For the problems no optimal solution is found)
- CPU time (in seconds)

For the construction heuristics we give the frequency each defines the best construction solution.

The heuristic results are compared with the heuristic algorithm of Hsu and Kim (2005). As discussed in Chapter 3, Hsu and Kim (2005) also consider renewable resources in their study, the only difference of their problem and our study is that they consider multiple resources.

All these performance measures are reported by their worst case (maximum) and average values.

We solve our mathematical models, and linear relaxations by GAMS using CPLEX solver. The algorithms are coded in C programming language. We conduct our experiments in an Intel Core(2) Duo 2.33 G.Hz, 1GB RAM computer.

6.3 Analysis of the Results

Our data set includes 60 instances of complexity 1.5 and 20 instances of complexity 2.1. In the tables below, we evaluate our performance measures with respect to the optimal objective function values for the problem with up to 30 activities. It is because 5 out of 10 instances are solved in more than 30 minutes by CPLEX and none of the 40 activity instances could be solved to optimality in 30 minutes.

We first investigate the effect of the mode elimination rules in reducing the problem size. Recall that Property 2 and Property 3 define the start times and modes of the activities whereas Properties 1, 4 and 5 eliminate the modes.

In Table 6.1, we report on Property 1. The table includes the number of activities, number of noncritical activities, total and average number of modes for noncritical activities, the total number of modes reduced by those assignments is given.

We also find the percentage of elimination in modes for these activities as $\%EL = \frac{Number\ of\ \ Modes\ Eliminated}{Number\ of\ \ Modes}$ and report the results.

Table 6-1 – The effect of Property 1 on the problem size

N	Number of noncritical activities		Total number of modes of noncritical activities		Average number of modes of noncritical activities		Number of activity modes reduced by Property 1		*%EL*	
	Avg.	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**
10	5.3	7	36	58	6.6	8.3	16.8	38	42.9	66
20	13	14	90.7	110	6.9	8.1	8.8	30	36.1	63
30	21.5	23	154.2	165	7.7	7.2	48.4	92	31.2	58
40	30.1	32	215.1	235	7.2	8.2	55.4	99.3	25.2	43
60	48	52	347.8	377	7.2	7.6	69.2	117	19.8	31
80	67.2	70	483.8	510	7.2	7.5	107.3	175	22.2	38

It is seen from the table that Property 1 performs better with smaller problems as expected. Consider $N=10$ and $N=20$, on the average 39% of the modes are eliminated by Property 1 in these problem sizes whereas for $N=60$ and $N=80$ on the average 21% of the modes are eliminated. This is due to the fact that the makespan for smaller problems is more restrictive, so that the activities tend to have fewer slacks. Also it can be deducted from the table that Property 1 has an influence of 27.5% reduction in the number of modes on the average. It is clear that reducing the number of modes by 27.5% is an important reduction in the problem size, verifying the effectiveness of Property 1.

In the algorithm we first use Property 1 as we do not need any lower bound for it. Then, for the reduced problem and with a good lower bound on hand we use Property 2, to improve the lower bound and decide on some activities' starting time and mode assignments.

Table 6.2 reports on the efficiency of Property 2 and Property 3. We let N_R denote the set of activities remained after employing Properties 2 and 3. In the table, we report the number of activities, number of noncritical activities, $|N_R|$ i.e. total number of activities in N_R, number of modes per activity over set N_R, total number of modes eliminated by Property 3.

Table 6-2 – The effect of Property 2 and Property 3

N	Noncritical activities		$\lvert N_R \rvert$		Total number of modes for N_R		Mode /activity for N_R		Total number of modes eliminated by Properties 2 and 3		*% EL*	
	Avg.	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**
10	5.3	7	4.4	6	17.1	27	3.99	5.4	3	10	0.15	0.47
20	13	14	10.7	13	45.8	63	4.18	5.25	8.8	30	0.17	0.65
30	21.5	23	19.4	21	97.1	118	5.01	6.56	8.9	14	0.09	0.21
40	30.1	32	27.9	31	149.7	178	5.38	6.59	9.8	25	0.06	0.16
60	48	52	45.5	48	265.7	297	5.85	6.56	12.9	20	0.05	0.07
80	67.2	70	61.9	65	344.7	403	5.56	6.21	27.7	56	0.08	0.15

Note that when $N = 10$, about half of the activities are set critical and taken out of consideration. For the remaining activities, on average 1 out of 5 of them are fixed by either Property 2 or Property 3. When $N = 80$, the average number of noncritical activities is 67, 5 of which are fixed by the properties. It can be inferred from Table 1 that, for all problem sizes on the average, more than 7% of the activities' start time and mode decisions are given by Property 2 or Property 3 and over 6% of the modes of the noncritical activities are eliminated from further consideration. With each activity eliminated from the decision process, more than 4 modes of each activity on the average are reduced. Recall that the problem is NP-Hard hence reducing the number of activities is important and these two properties are quite effective in reducing the problem size.

As shown with the example, property 4 definitely cannot make eliminations due to the mode generation scheme. After eliminating many modes and finding the LB, we may use Property 5, but as Property 1, 2 and 3 all eliminate longer modes like Property 5, performing Property 1,2 and 3 before, all long modes are eliminated and Property 5 can eliminate 18 modes in total for all the problem sets.

Having discussed the effect of problem size reductions, we next study the performance of our lower bounds. We compare the performances relative to the optimal objective function values and use the following performance measure for LB_i,

$(\%dev)_i = \frac{OPT - LB_i}{OPT} * 100$, where

LB_i is the maximum resource consumption of lower bound i and

OPT is the optimal maximum resource consumption.

In Table 6.3, the average and maximum percent deviations from the optimal and the number of times lower bound returns the optimal solution are reported.

Table 6-3 – The relative deviations of the lower bounds from the optimal

N	*%dev for LB_1*			*%dev for LB_2*			*%dev for LB_3*			*%dev for LB_4*		
	Avg.	**Max.**	**Freq.**	**Avg.**	**Max.**	**Freq.**	**Avg.**	**Max.**	**Freq.**	**Avg.**	**Max.**	**Freq.**
10	22.8	38	0	13.6	32	1	12.3	25	0	5.7	11	2
20	35.3	52	0	21.6	36	0	11.5	22	0	4.6	9	2
30	48.0	58	0	44.6	68	0	10.2	23	0	7.0	12	0

LB_1 is the lower bound that performs the worst. For $N = 10$ it is 23% far from the optimal, this percentage increases even more when the problem gets larger. This is not surprising as this lower bound makes use of only two activities, and while the problem gets larger, the bound coming from the overlapping of two activities is relatively small. The same situation is valid also for LB_2 Its deviation from optimal is 14% for $N = 10$ and 44.6% for $N = 30$. Although the bound is not restricted to two activities, as the makespan becomes larger for larger problem instances, the activities have more slack and they are not restricted to any time intervals, i.e., their earliest completion times are earlier than their late start times.

LB_3 is a promising lower bound, as the number of activities in the problem instance increases, the lower bound performs better. On the average it is 11.3% away from the optimal solutions. As expected LB_4 is the best performing lower bound and it works consistently well over all problem sizes. The average deviation is below 7%, the maximum deviation is 12% and it finds optimal solutions in 4 out of 30 instances. We expect this satisfactory behavior of LB_4, as we support the *LP* relaxation with some valid cuts and Property 2 and Property 3, and make it stronger.

We find that the optimal objective function values are so small that a small absolute deviation from the optimal solution may lead to high relative deviation, as it is a ratio of the optimal solution. As the relative deviation of lower bounds from the optimal is valuable information to only some extent, we also report the absolute deviations of the lower bounds from the optimal objective function values in Table 6.4.

Table 6.4 reports the average optimal objective function value, average and maximum absolute deviations of each lower bound from optimal solution and average objective function values for lower bounds.

$Absdev_i = OPT - LB_i$, where OPT is the optimal objective function value and LB_i is the objective function value for Lower Bound i.

Table 6-4 - The absolute deviations of the lower bounds from optimals

N	OPT	$Absdev_1$			$Absdev_2$			$Absdev_3$			$Absdev_4$		
	Avg.	Avg.	Max.	Obj.	Avg.	Max.	Obj.	Avg.	Max.	Obj.	Avg.	Max.	Obj.
10	21.7	5.4	12	16.3	2.6	6	19.1	2.8	6	18.9	1.1	2	20.6
20	27.1	10.1	17	17	5.8	11	21.3	3.2	7	23.9	1.2	2	25.9
30	32	15.4	21	16.6	14.1	21	17.9	3.3	9	28.7	2.2	3	29.8
40	-	-	-	15.1	-	-	17.0	-	-	32.7	-	-	32.6
60	-	-	-	13.5	-	-	15.1	-	-	41.1	-	-	41.2
80	-	-	-	16.4	-	-	20.2	-	-	51.4	-	-	55.4

Note from Table 6.4 that the average relative deviation of 5.7% for LB_4 stems from an average absolute deviation of 1.1 units. That means; the lower bound is approximately one unit lower than the optimal solution value. It is better seen in Table 6.4 that, lower bounds LB_3 and LB_4 outperform LB_1 and LB_2. It is clear from tables 3 and 4 that for up to 30 activities, LB_4 performs better than LB_3. Table 6.4 shows that when $N = 40$ and $N = 60$, the objective function values for LB_3 and LB_4 are almost equal. When N becomes 80, LB_4 again performs better.

Table 6.5 reports the number of times each lower bound is best for each problem size.

Table 6-5 – The number of times each lower bound is the best

N	LB_1	LB_2	LB3	LB_4
10	2	3	5	10
20	0	0	1	10
30	0	0	5	8
40	0	0	6	5
60	0	0	6	6
80	0	0	1	9

The numbers include the ties hence the sum of the numbers in each row may be greater than 10.

The table illustrates that the first and second lower bounds never define the best one when the number of activities is more than 10. This is in the line with the deviation results. For $N = 40$ the third lower bound, LB_3, defines the best solution more frequent than LP Relaxation bound, LB_4. It finds the best solution in 6 out of 10 instances whereas the LP Relaxation bound finds the best solution in 5 out of 10 instances. When $N = 60$, LB_3 and LB_4 define the best solution with the same frequency of 6. However; when $N = 80$, the performance of LB_4 in defining the best solution becomes dominant. It finds the best lower bound in 9 out of 10 instances. This number is only 1 for LB_3.

We can conclude from tables 6.3, 6.4 and 6.5 that, in general, the lower bounds found by the first and second approaches are not satisfactory, and are inferior to the third and fourth lower bounds. The performance of the third lower bound, making use of minimum possible resource consumption idea, deteriorates as *N* gets larger. The strengthened *LP* Relaxation behaves consistently well over all problem instances and is far superior to other lower bounds.

Table 6.6 reports the CPU times of finding LB_4 and the optimal solution. We do not give the CPU times of other lower bounds as they are negligibly small.

Table 6-6 – The CPLEX time for the optimal and the lower bound

N	LB_4 (in seconds)		OPT (in seconds)	
	Avg	**Max**	**Avg**	**Max**
10	0.67	0.78	0.26	0.38
20	1.02	1.33	1.57	5.59
30	2.10	3.36	15083.66	108106.6
40	6.99	18.09	-	-
60	14.02	26.59	-	-
80	21.75	36.92	-	-

Recall that our problem is strongly NP-Hard hence one should expect that the solution times increase exponentially with problem size. The results in Table 6.6 verify these expectations. For small problem sizes, i.e., for $N = 10$ and $N = 20$, the average CPLEX times are less than 2 seconds. When N becomes 30 the solution times reach to 4 hours. For $N = 30$, the maximum time used to find an optimal solution is more than 30 hours although one of the instances is solved in less than a second. This follows the inconsistent behavior of optimal model solutions. The LP relaxation time also increases with the problem size, however not exponentially. The increases are linear, when N is increased from 40 to 80, i.e., 2 times, the average CPU times increases from 6.99 to 21.75 second, i.e., 3 times. Even when $N = 80$, the computation times do not reach to 1 minute.

Due to the satisfactory performance of the lower bounds, we use them to evaluate our heuristic procedure. In doing so, for each problem instance we select the lower bound giving the maximum value. Table 6.7 gives the objective function value of optimal, relative and absolute percent deviations of the best lower bound from the optimal solution and frequency of finding the optimal solutions. As before, we use $\%dev = \frac{OPT - LB}{OPT} * 100$ where $LB = \underset{i=1,2,3,4}{Max}\ LB_i$ for percent relative deviations. Accordingly the absolute deviations are measured as $Absdev = OPT - LB$.

Table 6-7 – The deviations of the best lower bound from the optimal

N	*OPT*		*Absdev*		*%dev*		**Frequency of optimal**
	Avg.	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**	
10	21.7	32	1.1	2	5.7	11	2
20	27.1	34	1.2	2	4.6	9	2
30	32	40	2	3	6.5	12	0

Table 6.7 illustrates that for $N = 10$ and $N = 20$ the best lower bound returns 4 optimal solutions out of 20 instances. When N becomes 30, the best lower bound cannot find the optimal solution which is a signal of deteriorating performances of the lower bounds with increases in the problem sizes. The deviations verify this conclusion. As seen from Table 6.7, the best lower bound is at most 3 units less than the optimal objective function value for all 30 problem instances. The average percent deviation is about 6%, which is quite good for estimating the optimal solution.

Having discussed the effect of mode elimination procedures and performance of lower bounds, we now discuss the performance of our heuristic procedure. As discussed in Chapter 5, our heuristic has two main phases; construction and improvement. In the construction phase, 3 different construction heuristics are used, namely Early Start Schedule *(ESS)*, Late Start Schedule *(LSS)*, Alternating Early-Late Start Schedule *(AELSS)*. We first measure their performance relative to the optimal solutions for small-sized problem instances up to 30 activities and relative to the best lower bound over all problem sets.

The relative deviations from the optimal solution and best lower bound are measured as follows;

i. Relative Deviation of CS_i from Optimal: $\%devopt = \frac{CS_i - OPT}{OPT} * 100$

where CS_i is the maximum resource usage of construction schedule i, where $i = ESS, LSS, AELSS$

ii. Relative Deviation of CS_i from Best Lower Bound: $\%gap = \frac{CS_i - LB}{LB} * 100$

where CS_i is the maximum resource usage of construction schedule i, where $i = ESS, LSS, AELSS$

Tables 6.8 and 6.9 report the average and maximum relative deviations from the optimal and the best lower bounds, respectively. Table 6.8 also reports number of instances out of 10 each construction heuristic returns the optimal solution.

Table 6-8 – The relative deviations of construction heuristics from the optimal

N	*%devopt for CS_1*		*OPT*	*%devopt for CS_2*		*OPT*	*%devopt for CS_3*		*Optimal*
	Avg.	**Max.**	**Freq.**	**Avg.**	**Max.**	**Freq.**	**Avg.**	**Max.**	**Freq.**
10	31.8	60	0	28.5	85	1	22	37	1
20	56.5	120	0	51.2	136	0	45.6	105	1
30	60.8	104	0	54.1	81	0	50.4	85	0

Table 6-9 – The relative deviations of construction heuristics from the best lower bound

N	*%gap for CS_1*		*%gap for CS_2*		*%gap for CS_3*	
	Avg.	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**
10	40.1	78	36.9	106	29.6	53
20	64.1	132	59.2	160	53	116
30	72.1	130	64.9	104	61	109
40	80.7	126	82.5	137	64.7	107
60	99.3	129	95.3	128	60.9	87
80	80.1	117	105.4	143	72	104

As it can be seen from Table 6.8, the construction heuristics return optimal solutions for small-sized problems with 10 and 20 activities. Although for some small-sized instances the construction heuristics return very good results, their average performance is not as good. Even for $N = 10$, the construction heuristics return solutions that are 20-30% away from the optimal solution on average. This is

not surprising as even the most complex construction schedule *(AELSS)* only considers early or late start times for the start times of the activities using a specified mode. Table 6.8 also reveals that when the problem size gets larger, the performance of the construction heuristic deteriorates. For $N = 30$, the best construction heuristic *(AELSS)* generates solutions with objective function values over 50% far from the optimal. So it is clear that the performance of these construction heuristics is not satisfactory and needs to be improved. Table 6.9 gives the deviations from the best lower bound. This table clearly shows that *AELSS* is by far the best construction heuristic as it performs better than others over all problem set.

We next study the performance of the best construction heuristic. The performances are measured by the absolute deviations from the optimal solution and best lower bound that are stated below.

i. Absolute Deviation from the Optimal: $Absdevopt = \underset{i}{Min}\{CS_i\} - OPT$ where CS_i is the maximum resource usage of construction heuristic i, where $i = ESS, LSS, AELSS$

ii. Absolute Deviation from the Best Lower Bound: $Absgaplb = \underset{i}{Min}\{CS_i\} - LB$ where CS_i is the maximum resource usage of construction heuristic i, where $i = ESS, LSS, AELSS$

Table 6.10 and Table 6.11 report the relative and absolute deviations of the best construction heuristic from the optimal objective function value and the best lower bound, respectively. Table 6.11 also reports on the frequency of the best construction heuristic.

Table 6-10 – The absolute and relative deviation of the best construction heuristic from the optimal

N	***Absdevopt***		***%dev for best CS***	
	Avg.	**Max.**	**Avg.**	**Max.**
10	3.5	6	18.6	33
20	8.1	20	31.8	67
30	14.8	21	0.481	0.81

Table 6-11- The absolute and relative deviation of the best construction heuristic from the best lower bound and frequency of best construction heuristic

N	***Absgap***		***%gap for best CS***		***Best Schedule frequency***		
	Avg.	**Max.**	**Avg.**	**Max.**	**CS_1**	**CS_2**	**CS_3**
10	4.6	8	26.2	47	2	7	7
20	9.3	22	38.4	79	3	3	6
30	16.8	24	58.5	104	3	5	6
40	19.1	25	59.9	85	2	3	6
60	25.3	39	60.9	87	0	0	10
80	35.4	47	65.4	98	2	1	6

For each problem instance the best construction heuristic is chosen and its deviation from optimal is found. Table 6.10 illustrates these results. The table shows that the maximum deviations from optimal are about twice the average deviations. For instance when $N = 20$ the average deviation from the optimal objective function value is 8.1, the worse case brings a solution that deviates about 20 units from the optimal. This reveals that the construction heuristics' solutions are not consistent. Also, the average relative deviations of the best construction heuristic from the optimal are increasing consistently and sharply with the increasing number of activities.

It is seen from table 11 that, even selecting the best construction heuristic does not result in solutions close to the best lower bounds. For example when $N \geq 30$, the deviation of the best construction heuristic from the best lower bound is approximately 60%. It can be observed that the best construction heuristic's objective function value may even be twice of the best lower bound. Table 11 shows that *AELSS* is the best construction heuristic that deviates least from optimal. However from some instances the other construction heuristics give better results. *ESS* gives the best solutions in 12 out of 60 instances, *LSS* and *AELSS* give 19 and 41 instances respectively, including the ties. To summarize the majority of the best solutions come from *AELSS* and the other construction heuristics are also worth

consideration. The results on the construction heuristics' performances point a need for improvement.

We compare our heuristic results after the improvement step, not only with the optimal solutions and lower bounds but also with the heuristic of Hsu and Kim (2005) *(HKH)* as discussed in section 6.2. Both our heuristic and *HKH* are applied to the problem reduced by mode elimination mechanisms. The performances of both heuristics are reported in tables 6.12 through 6.15. In all the tables, the same performance measures are reported for both heuristics.

Table 6.12 and Table 6.13 report on the performance of both heuristics with respect to the optimal solution for the problem sets when $N \leq 30$. Table 6.12 reports the average and maximum relative deviation of the heuristics from the optimal solution and the number of times they find the optimal solution. Table 13 reports the average and maximum optimal objective function values and absolute deviation of both heuristics from optimal which are found as below.

i. Relative Deviation of UB_i from Optimal: $\%devopt = \frac{UB_i - OPT}{OPT} * 100$ where UB_i is the maximum resource usage of heuristic i where $i = IH, HKH$

ii. Absolute Deviation of UB_i from Optimal: $Absdevopt = \underset{i}{Min}\{UB_i\} - OPT$ where UB_i is the maximum resource usage of heuristic i where $= i = IH, HKH$

Table 6-12 – The relative deviations of the best upper bound from the optimal and frequency of optimal solutions

N	***%dev* for *IH***		**Frequency of optimal**	***%dev* for *HKH***		**Frequency of optimal**
	Avg.	**Max.**		**Avg.**	**Max.**	
10	1.6	10	8	16.5	28	0
20	5.4	9	1	20.3	32	0
30	6.2	9	0	29.4	44	0

Table 6.12 shows that *IH* provides 9 optimal solutions in 30 instances, 8 of which are for 10 activities. The average relative deviation from optimal is around 6 %. It is seen from the table that *HKH* cannot find any optimal solution for 30 problems; it does not even perform well even for $N = 10$. Actually, this is not surprising as that

algorithm is a priority based rule and it only positions the activities to anywhere in its allowable range and it does not use any improvement procedure.

Table 6-13 - The absolute deviations of the best upper bound from the optimal

N	***OPT***		***Absdev* for *IH***		***Absdev* for *HKH***	
	Avg.	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**
10	21.7	32	0.3	2	3.5	8
20	27.1	34	1.4	3	5.5	8
30	32	40	1.9	3	9.5	15

Table 6.13 reveals the satisfactory behavior of *IH*. The average absolute deviation of the heuristic from the optimal objective function value is less than 2 units, 1.9 units, for $N = 30$. Note that the maximum absolute deviation from the optimal objective function value is only 3, hence the heuristic behaves consistently well over all problem instances. *HKH* returns an average absolute deviation of 3.5 units for the smallest problem set that is greater than the maximum absolute deviation of *IH* for $N \leq 30$. These results are enough to conclude that for small problem instances whose optimal solutions are known, *HKH* is inferior to *IH*.

For the problem sets including more than 30 activities, we compare our results with the best lower bounds. Note that the deviation of the best lower bound from the optimal objective function value for $N \leq 30$ is around 6%.

We calculate the relative and the absolute deviations of the heuristics from the best lower bound available and report the results in Tables 6.14 and 6.15, respectively.

i. Relative Deviation of UB_i from the best lower bound: $\%gaplb = \frac{UB_i - LB}{LB} * 100$ where UB_i is the maximum resource usage of heuristic i where $i = IH, HKH$

ii. Absolute Deviation of UB_i from the best lower bound: $Absgaplb = \underset{i}{Min}\{UB_i\} - LB$ where UB_i is the maximum resource usage of heuristic i where $= i = IH, HKH$

Table 6.15 also includes the objective function values of the best lower bounds.

Table 6-14 – The relative deviations of the upper bound from the best lower bounds

N	***%gaplb* for *IH***		***%gaplb* for *HKH***	
	Avg.	**Max.**	**Avg.**	**Max.**
10	7.9	22.2	23.7	38.9
20	10.4	20	26	32.2
30	13.4	21.7	38.3	53.6
40	13.5	15	33.2	45.9
60	16.6	23	48.4	85.7
80	16.1	21	38.9	47.6

The average deviation of *IH* from the best lower bound is very small, 7.9%, for $N = 10$. It is due to the fact that, the lower bounds are very close to the optimal objective function values for that problem size and the upper bounds found by *IH* are nearly optimal. The same is true for $N = 20$. For larger problem sizes, we expect that the relative gap between the lower bound and upper bound increases. In line with our expectations, our experimental results show that our lower bounds deteriorate by the increase in the number of activities. Looking at the deviations in Table 6.14, it can be seen that the upper bounds of problem sets with more than 20 activities deviate more from the lower bound. But, the table shows that the maximum deviation from the best lower bound for *IH* is always less than 23% and the average deviation is around 15 %. *HKH*, when the problem size increases does not get better, but we can say that it does not get worse either. For $N = 80$, for example, the deviation from the best lower bound still deviates less than 40% from the best lower bound as for $N = 30$.

Table 6-15 – The absolute deviations of the upper bound from the best lower bounds for each heuristic

N	***LB***		***Absgap* for *IH***		***Absgap* for *HKH***	
	Avg.	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**
10	20.6	32	1.4	4	4.6	8
20	25.9	34	2.6	5	6.7	10
30	30	38	3.9	5	11.5	17
40	33.4	49	4.4	5	11.4	21
60	42	56	6.8	9	20.3	36
80	55.8	67	8.9	11	21.7	29

It should be noted from Table 6.15 that, as the problem size increases the objective function values also increase. The objective function values for $N = 80$ are twice more than those of $N = 20$ and the deviation from the best lower bound increases in the same way. Again, from Table 6.15 it can be seen that the maximum deviation of *IH*, from the best lower bound is less than half of the average deviation of *HKH* for over all problem set. All these results, and the fact that *HKH* could not perform better than *IH* for even a single instance, shows the superiority of *IH* over *HKH*.

Table 6.16 shows the average and maximum computational times of finding the heuristic algorithm (construction and improvement all together) and *HKH*. The mode elimination procedures are not reported as they use negligible time (less than one tenth of a second). All CPU times are reported in seconds.

Table 6-16 – The CPU times of the algorithms

N	CPLEX TIME		BEST LB		OUR HEURISTIC		*HKH*	
	Avg.	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**
10	0.264	0.38	0.52	0.63	0.94	1.27	0.53	0.65
20	2.24	5.04	0.99	1.33	2.28	4.50	1.02	1.42
30	15083.66	108106.6	1.86	3.2	4.84	8.42	2.13	3.72
40	-	-	3.64	6.72	9.20	20.02	4.36	6.88
60	-	-	15.02	32.92	30.94	52.92	20.56	40.95
80	-	-	23.26	48.33	48.86	86.67	35.63	65.77

As seen in Table 6.16, the CPLEX time increases exponentially with increases in the number of activities. On the other hand the lower bound times increase linearly. The lower bounds are used both by our heuristic and *HKH*, hence we include the lower bound times to all heuristic times. The CPU times spent both by our heuristic and *HKH* are quite small. Note from the table that the maximum time for the highest number of activities, i.e., $N = 80$, is slightly less than 90 seconds by our algorithm and 70 seconds by *HKH*. For all problem instances the *HKH* produces quicker solutions than our heuristic however at an expense of lower quality.

As discussed before, we used 20 instances with *CNC* value of 2.1 having 20 and 80 activities, in the following tables we will denote them *20+* and *80+* respectively. Recall that, Alvarez and Tamarit (1989) discuss that when *CNC* increases the problem becomes easier. This makes sense because the starting time alternatives for activities decrease as the network becomes more interrelated.

Then, we discuss the effect of the network complexity on the speed and quality of the solutions. We extend our experiment with 20 instances of 20 and 80 activities each having *CNC* value of 2.1. We compare the results with those of the same size with CNC value of 1.5. As discussed by Alvarez and Tamarit (1989) when the *CNC* value increases the problems become easier to solve. Hence our main experiment with small CNC values includes hard-to-solve problem instances.

We analyze the effect of the *CNC* value in the reducing the problem size and report the results in Table 6.17. The table includes the average and maximum number of

noncritical activities, number of activities in the set N_R, total number of modes of noncritical activities, number of modes eliminated by Property 1, number of modes eliminated by Property 2 and 3, and total percent reduction.

Table 6-17 – The problem size comparison for different *CNC* values

N	*CNC*	**Noncritical Activities**		N_R		**Total number of modes for noncritical activities**		**Number of Modes Eliminated by Property 1**		**Number of Modes Eliminated by Property 2 and 3**		*%EL*	
		Avg.	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**
20	1.5	13	14	10.7	13	90.7	110	36.2	63	8.8	30	48.3	83.8
	2.1	12.5	14	11.5	13	87.3	108	34.3	53	3.3	11	42.2	62.8
80	1.5	67.2	70	62	65	483.8	510	107.2	175	27.3	56	27.8	46.6
	2.1	65.9	70	60.8	68	477.6	509	137.4	204	22.4	35	33.2	49.2

Note from the table that for small *N*, *N* = 20, the power of the elimination for different *CNC* values are very close. When N = 80, the effect of property 1 becomes more visible. The property eliminates about 5% more modes when the CNC value increases from 1.5 to 2.1. This is due to having tighter precedence relations that leave smaller room for the start times.

The performance of the best lower bound, LB_4, relative to the optimal solution is evaluated only for $N = 20$ as the optimal solutions are not available for N = 80. When $CNC = 2.1$, the deviation of the best lower bound from the optimal is 4.8% hence is better than that of $CNC = 1.5$ which was found around 6%. This is due to the fact that when the CNC value is higher, there are more precedence constraints and the total slack values of the activities are smaller. This leaves fewer choices for the decision variables; hence the resulting solutions are more close to the optimal ones.

The performances of our heuristic and *HKH* are given in Table 6.18. The table reports the number of activities and average and maximum relative deviations of heuristics *IH* and *HKH* from the optimal solution (for $N = 20$) and lower bounds for each *CNC* value.

Table 6-18 - The deviations of heuristics for different *CNC* values

N	***CNC***	***%dev* for *IH***		***%dev* for *HKH***		***%gap* for *IH***		***%gap* for *HKH***	
		Avg.	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**
20	**1.5**	5	12	23.1	31	10.4	19	29.3	40
	2.1	5.4	9	20.3	32	5.6	10	26.1	32
80	**1.5**	-	-	-	-	16.1	21	38.9	48
	2.1	-	-	-	-	15.5	25	46	62

As can be observed from Table 6.18, there is no significant and consistent effect of CNC on the quality of the heuristic procedures. The percentage deviations of different CNC values are very small and close. Note that the average relative deviation of *IH* increases from 5% to 5.4%, when the *CNC* value increases from 1.5 to 2.1. On the other hand, the average relative deviation of *HKH* decreases from 23.1% to 20.3%, when the CNC value increases from 1.5 to 2.1. The similar results hold for the percentage gaps.

Finally, we discuss the effect of the CNC values on the solution times. Table 6.19 reports the CPU times for the optimal solutions, best lower bound, *IH* and *HKH* heuristics.

Table 6-19 – The CPU time comparison for different *CNC* values

N	***CNC***	**CPLEX Time**		**CPU Time of Best LB**		**CPU Time of *IH***		**CPU Time of *HKH***	
		Avg.	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**
20	1.5	2.24	5.04	0.99	1.33	2.28	4.50	1.02	1.42
	2.1	0.48	0.69	1.02	1.36	2.15	3.07	1.04	1.39
80	1.5	-	-	23.26	48.33	48.86	86.67	35.63	65.77
	2.1			30.19	68.42	52.65	115.64	42.01	92.47

Note from the table that the average CPU time is 2.24 seconds when CNC = 1.5 and 0.48 seconds when *CNC* = 2.1. This is due to the fact that when *CNC* is higher, there are more precedence constraints and activity slack times are smaller. This follows the binary decision variables are fewer, hence the optimal solutions can be

obtained quicker. For the best lower bound and heuristic procedures we do not observe any significant effect of the *CNC* values on the performances. There is a slight increase in CPU times which can be attributed to the increased project completion time due to the added precedence constraints.

We now discuss the effect of the number of modes. Table 6.20 illustrates the effect of number of modes on the problem size reduction mechanisms. In the table, together with number of activities we illustrate number of non-critical activities, $|N_R|$, total number of modes for N_R, number of modes eliminated by properties, and total reduction in problem size by the reduction mechanisms.

Table 6-20 – The problem size comparison for different number of modes settings

N	***Mode Setting***	**Noncritical Activities**		N_R		**Total number of modes for noncritical activities**		**Number of Modes Eliminated by Property 1**		**Number of Modes Eliminated by Property 2 and 3**		***%EL***	
		Avg.	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**
20	1	13	14	10.4	13	49.8	61	18.5	35	6.3	21	48.1	90
	2	13	14	10.7	13	90.7	110	36.2	63	8.8	30	48.3	83.8
40	1	30.1	32	27.9	31	117.1	129	27.6	52	5.6	14	27.6	51.7
	2	30.1	32	27.9	31	300.7	344	76.4	134	9.8	25	28.4	45.6

Note from Table 6-20 that the number of noncritical activities does not change as the smallest duration modes do not change due to the mode generation scheme. It is clear from the total problem size reduction percentages that the number of modes has no effect on the elimination power of the properties.

Table 6-21 illustrates the effect of number of modes on the performance of our heuristic procedure. It reports percent deviation of the *IH* and *HKH* solutions from the optimal solution and the best lower bound value.

Table 6-21 - The deviations of heuristics for different number of modes settings

N	*Mode Setting*	*%dev* for *IH*		*%dev* for *HKH*		*%gap* for *IH*		*%gap* for *HKH*	
		Avg.	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**
20	1	3.8	10	18.3	24	11.3	25	26.6	32
	2	5	12	23.1	31	10.4	19	29.3	40
40	1	-	-	-	-	13.2	15	35.2	52
	2	-	-	-	-	13.5	15	33.2	45.9

According to Table 6-21, when there are fewer modes, the heuristic solutions are closer to the optimal solution. When compared with the lower bounds, the performances are similar under both settings. For both mode settings, our heuristic outperforms *HKH*.

Table 6-22 reports the CPU times for the optimal solution, best lower bound, and the heuristics, *IH* and *HKH*.

Table 6-22 - The CPU time comparison for different mode number settings

N	*Mode Setting*	CPLEX Time		CPU Time of Best LB		CPU Time of *IH*		CPU Time of *HKH*	
		Avg.	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**	**Avg.**	**Max.**
20	1	1.33	2.34	0.71	0.97	1.65	2.44	0.72	1.02
	2	2.24	5.04	0.99	1.33	2.28	4.5	1.02	1.42
40	1	-	-	2.34	3.78	7.7	13.67	2.9	4.58
	2	-	-	3.64	6.72	9.31	18.02	4.36	7.7

Table 6-22 reveals that not only the optimal solutions but also the bounding procedures are found quicker when there are fewer modes. However the effect of the number of activities is more dominant than that of number of modes, for all approaches. Note that when N increases two times from 20 to 40, the CPU times of the best lower bound increase from 0.71 to 2.34 seconds, respectively, for the fewer mode case. On the other hand, when the number of modes increase two times, the CPU times of the best lower bound increase from 0.71 to 0.99 seconds.

CHAPTER 7

CONCLUSIONS

In this study, we consider a resource investment problem with time-resource trade-offs. We assume each activity consumes a renewable resource during its execution and the duration of an activity can be reduced by consuming extra resources. Each time- resource combination defines a mode for an activity and there are several activity modes. We constrain the project length by the completion time found when all activities are set to their minimum duration modes.

Our problem to find activity durations that minimizes the maximum resource without increasing the minimum project length. The problem is shown to be strongly NP-hard. Hence optimal solutions to the medium to large sized instances are hardly obtained in reasonable time.

In this study we propose several problem size reduction techniques, heuristic and the lower bounding procedures. The results of our experiments have revealed that the reduction techniques and bounding procedure perform very satisfactory over all problem set. The performances slightly deteriorate with the increases in the number of activities, number of modes and decreases in the complexity index. Even for the largest problem sizes, our heuristic solutions deviate from the best lower bounds by about 16%.

We observe that our mode elimination procedures reduce the problem size by more than 30%, either by eliminating the modes of the activities or settling their start

times. Lower bounds serve as underestimates on the optimal objective function values. We find that our lower bounds produce solutions that are very close to optimal. For problem sizes with less than 40 tasks, the average relative deviation of the best lower bound from the optimal is around 6% and the absolute deviation of the best lower bound from the optimal is no more than 3 resource units.

Our heuristic procedure runs in two steps: construction and improvement. We find that our heuristic's performance is very similar to that of our lower bound as it deviates from the optimal by 6% on the average and the average absolute deviation from the optimal is again no more than 3 units. We also compare our heuristic with the heuristic proposed by Hsu and Kim (2005) and find that our heuristic is far superior. It has outperformed Hsu and Kim (2005)'s heuristic in all instances without any exception.

We hope that our study fills an important gap in the resource constrained project scheduling literature. To the best of our knowledge there is a unique reported study that tackles with our problem. The study proposes a heuristic solution that is found to have inferior performance when compared with ours.

The future research around our problem may consider the following issues. Our reduction mechanisms and bounding approaches can be embedded into an optimization procedure. Our results can be extended to the preemptive activities case. We assume a single renewable resource; future research may consider multiple renewable resources and nonrenewable resources as well. Other resource leveling objectives like minimizing range of resource usages may be worth studying. Three attribute trade-offs including time, cost and resource can also be considered in future research.

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*, Vol.165, No.2, pp. 339-358

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

Alvarez-Valdes, R. and J.M. Tamarit (1989): Heuristic algorithms for resource constrained project scheduling: A review and an empirical analysis. In: Slowinski, R. and J. Weglarz (Eds.): *Advances in project scheduling. Elsevier, Amsterdam*, pp. 113-134

Antonio F. Gómez-Skarmeta, Fernando Jiménez and Jesús Ibáñez (1999), Pareto-optimality in Scheduling, *Problems Lecture Notes in Computer Science*, 1625, pp. 177-185

De Reyck B., Demeulemeester E., Herroelen W. (1998), Local search methods for the discrete time/resource trade-off problem in project networks, *Naval Research Logistic Quarterly*, 45, pp. 553–578.

Değirmenci, G., (2008) The Budget Constrained Discrete Time/Resource Trade-off Problem in Project Networks, MSc Thesis, METU IE.

Demeulemeester E., De Reyck B., Herroelen W. (2000), The discrete time/resource trade-off problem in project networks: a branch and bound approach, *IIE Transactions*, 32, pp. 1059–1069.

Demeulemeester, E. (1995), Minimizing Resource Availability Costs in Time Limited Project Networks, *Management Science*, 41, pp. 1590-1598

Drexl, A. and Kimms, A. (2001), Optimization Guided Lower and Upper Bounds for the Resource Investment Problem, *Journal of the Operational Research Society*, 52, pp. 340-351.

Drexl, A., and Gruenewald, J. (1993), Nonpreemptive multi-mode resource-constrained project scheduling, *IIE Transactions*, 25, pp. 74-81.

Hartmann S (2001). Project scheduling with multiple modes: a genetic algorithm, *Annals of Operations Research*, 102, pp. 111-135.

Hartmann, S., Briskorn, D. (2009), A survey of variants and extensions of the resourceconstrained project scheduling problem, *European Journal of Operational Research*, Vol. 207, No. 1, pp 1-14

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

Hsu, C. C. and Kim, D. S. (2005) A new heuristic for the multi-mode resource investment problem. *Journal of the Operational Research Society*, **56**, pp. 406-413.

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

Kelley Jr., J.E., (1963), The critical-path method: Resources planning and scheduling, *Industrial Scheduling Prentice-Hall, New Jersey*, pp. 347–365.

Kolisch, R. and Sprecher A. (1996): PSPLIB - A project scheduling library, *European Journal of Operational Research*, Vol. 96, pp. 205--216.

Meredith, M. (2003): Project Management: A Managerial Approach, 5th Edition

Möhring R. (1984), Minimizing costs of resource requirements in project networks subject to a fixed completion time, *Operations Research*, 32, pp. 89–120

Mori M and Tseng CC (1997), A genetic algorithm for multi- mode resource constrained project scheduling problem, *European Journal of Operational Research*, 100, pp. 134-141.

Nudtasomboon N., Randhawa S.U. (1997), Resource-constrained project scheduling with renewable and non-renewable resources and time-resource tradeoffs, *Computers and Industrial Engineering*, 32 (1), pp. 227–242.

Peteghem, V. and Vanhoucke, M. (2009), A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem, *European Journal of Operational Research*.

Pulat PS, Horn SJ (1996) Time–resource tradeoff problem. *IEEE Transactions on Engineering Management*, 43(4), pp. 411–417

Ranjbar M., De Reyck B., Kianfar F. (2009), A hybrid scatter search for the discrete time/resource trade-off problem in project scheduling, *European Journal of Operational Research*, 193, pp. 35–48.

Ranjbar M., Kianfar F. (2007), Solving the discrete time/resource trade-off problem in project scheduling with genetic algorithms, Applied Mathematics and Computation 191, pp. 451–456.

Ranjbar M., Kianfar F., Shadrokh S.(2008), Solving the resource availability cost problem in project scheduling by path relinking and genetic algorithm, *Applied Mathematics and Computation*, 196, pp. 879–888.

Shadrokh S., Kianfar F. (2007), A genetic algorithm for resource investment project scheduling problem, tardiness permitted with penalty, *European Journal of Operational Research*, 181 (1), pp. 86–101.

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

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

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

Valls V., Quintanilla M.S., Ballestin F. (2003), Resource-constrained project scheduling: a critical reordering heuristic, *European Journal of Operational Research*, 165, pp. 375–386.

Yamashita D.S., Armentano V.A., Laguna M. (2007), Robust optimization models for project scheduling with resource availability cost, *Journal of Scheduling*, 10(1), pp. 67–76.