AN EFFICIENT BRANCH AND BOUND ALGORITHM FOR THE RESOURCE LEVELING PROBLEM

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

BY

HÜSEYİN YENİOCAK

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

AUGUST 2013

Approval of the thesis:

AN EFFICIENT BRANCH AND BOUND ALGORITHM FOR THE RESOURCE LEVELING PROBLEM

submitted by **HÜSEYİN YENİOCAK** in partial fulfillment of the requirements for the degree of **Master of Science in Civil Engineering Department, Middle East Technical University** by,

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

Prof. Dr. Ahmet Cevdet Yalçıner _____
Head of Department, **Civil Engineering**

Assoc. Prof. Dr. Rıfat Sönmez _____
Supervisor, **Civil Engineering Dept., METU**

Asst. Prof. Dr. Tankut Atan _____
Co-Supervisor, **Industrial Engineering Dept., Işık University**

Examining Committee Members:

Asst. Prof. Dr. Metin Arıkan _____
Civil Engineering Dept., METU

Assoc. Prof. Dr. Rıfat Sönmez _____
Civil Engineering Dept., METU

Asst. Prof. Dr. Tankut Atan _____
Industrial Engineering Dept., Işık University

Prof. Dr. M. Talat Birgönül _____
Civil Engineering Dept., METU

Asst. Prof. Dr. Aslı Akçamete _____
Civil Engineering Dept., METU

Date: 05/08/2013

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 : Hüseyin Yeniocak

Signature :

ABSTRACT

AN EFFICIENT BRANCH AND BOUND ALGORITHM FOR THE RESOURCE LEVELING PROBLEM

Yeniocak, Hüseyin
M.Sc., Department of Civil Engineering
Supervisor : Assoc. Prof. Dr. Rıfat Sönmez
Co-Supervisor : Asst. Prof. Dr. Tankut Atan

August 2013, 92 Pages

Resource leveling problem (RLP) focuses on optimizing resource utilization curves obtained by using Critical Path Method (CPM). This thesis presents a new and efficient branch and bound algorithm for the RLP. The proposed algorithm has been compared with mixed-integer linear modeling methods and previous branch and bound algorithms. An adaptive branch and bound heuristic has been developed for a good upper bound calculation. A new lower bound calculation strategy has been introduced and dual calculation has been used to obtain better lower bound values. Activity selection methods for branching process have been proposed to improve lower bounds and accelerating techniques have been implemented to achieve an efficient branch and bound algorithm for the RLP.

Extensive computational experiments have been conducted using test problems from the literature including instances from Project Scheduling Problem Library (PSPLIB), Kolisch et al. (1999), Franck et al. (2001) and Rieck et al. (2012). The branch and bound algorithm has proven the best performance in terms of computational times for problems up to 20 activities for all objective functions. Moreover, for resource idle days optimization, problems up to 30 activities were solved for the first time in literature. Mixed-integer linear model could only solve problems with 10 activities for resource idle days and its performance highly depends on the type of the objective function. The proposed branch and bound algorithm provides a powerful method for the RLP due to its flexibility in applying several accelerating techniques and it can solve optimization functions in all forms.

Keywords: Resource Leveling Problem, Branch and Bound, Lower Bound, Mixed-Integer Linear Modeling, Resource Idle Days

ÖZ

KAYNAK DENGELEME PROBLEMİ İÇİN ETKİN BİR DAL VE SINIR ALGORİTMASI

Yeniocak, Hüseyin
Yüksek Lisans, İnşaat Mühendisliği Bölümü
Tez Yöneticisi : Doç. Dr. Rıfat Sönmez
Ortak Tez Yöneticisi : Yard. Doç. Dr. Tankut Atan

Ağustos 2013, 92 Sayfa

Kaynak dengeleme problemi (KDP) projelerin kaynak kullanım eğrilerini en iyilemeyi amaçlamaktadır. Bu kapsamda, dal ve sınır tabanlı bir algoritma geliştirilmiştir. Geliştirilen dal ve sınır algoritması doğrusal karma-tamsayılı modelleme yöntemi ile literatürde yer alan diğer dal ve sınır tabanlı yöntemlerle karşılaştırılmıştır. İyi bir üst sınır değeri hesabı için uyarlamalı bir dal ve sınır sezgiseli geliştirilmiştir. İkinci bir alt sınır değeri hesabı sunularak ikili alt sınır değeri hesabı KDP'nin çözümünde ilk defa kullanılmıştır. Alt sınır değerlerini geliştirmek için dal açma işleminde kullanılmak üzere çeşitli aktivite seçim metodları sunulmuştur. Algoritmayı hızlandırmak ve yüksek performanslı bir yöntem elde etmek amacıyla çeşitli teknikler uygulanmıştır.

İçerisinde PSPLIB, Kolisch vd. (1999), Franck vd. (2001) ve Rieck vd. (2012)'den problem örneklerinin de olduğu literatürden alınan test setleri kullanılarak kapsamlı deneyler yapılmıştır. Geliştirilen dal ve sınır algoritması 20 aktiviteye kadar olan problemlerin tüm amaç fonksyonlarına göre çözümünde işlem süresi bakımından literatürdeki en iyi performansı sergilemiştir. Ayrıca, atıl kaynak günü en iyilemesinde literatürde ilk defa, dal ve sınır algoritması 30 aktiviteye kadar olan problemleri çözmüştür. Doğrusal karma-tamsayılı model aynı amaç fonksyonu için 10 aktiviteye kadar olan problemleri çözebilmiş ve bu yöntemin performansının kullanılan amaç fonksiyonuna yüksek oranda bağımlı olduğu gözlenmiştir. Geliştirilen dal ve sınır algoritması KDP'nin çözümünde işlem süresini hızlandırma tekniklerinin uygulanabilmesindeki ve birbirinden farklı amaç fonksiyonlarını çözebilmesindeki esnekliğiyle en iyileme algoritmaları için güçlü bir yöntem sunmaktadır.

Anahtar Kelimeler: Kaynak Dengeleme Problemi, Dal ve Sınır Yöntemi, Alt Sınır, Doğrusal Karma-Tamsayılı Modelleme, Atıl Kaynak Günü

To My Most Beloved One

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Assoc. Prof. Dr. Rıfat Sönmez for his support, encouragement, comments and critiques on this work he provided at any time. I am very thankful to him not only for appreciating me to study at his Research and Development project founded by TÜBİTAK but also for his support and guidance for my future work life decisions.

I would like to thank my co-research assistant, PhD candidate Mahdi Abbasi Iranagh for his comments, critiques, and share of his experience. He shared with me the same office room through all my study at Civil Engineering Department. He helped me in solving many issues of my work and always responded when I needed help without hesitation.

I also want to acknowledge thanks to my friend Emad Rezvankhah for his help and effort in doing our assignments. He showed great patience when I was stack on my study and could not perform my duty in our assignments immediately.

I would like to thank my co-advisor, Asst. Prof. Dr. Tankut Atan for his effort on devising the part of this thesis study related to the mixed-integer linear modeling.

The financial support of TÜBİTAK to the Research and Development Project of 111M140 is gratefully acknowledged. Without the financial support it was difficult for us to provide the utilized equipment and materials for the study and even to afford living in Ankara.

I also want to express my great thanks to my girlfriend Kevser Yaman, who has become my source of motivation during this study and through all my life. She constituted the essential part of my life by providing sensational help at every minute. It would have been very difficult for me to success my study without her.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

ABSDEV	Absolute Deviation Metric
AoN	Activity on Node
B&B	Branch and Bound
CPM	Critical Path Method
DSS	Decision Support System
DUR	Duration
EF	Early Finish Time
ES	Early Start Time
FS	Finish to Start
GA	Genetic Algorithm
GASA	Genetic Algortihm with Simulated Annealing
GB	Gigabyte
GHz	Gigahertz
ID	Identity
lb	lower bound
LBC	Lower Bound Calculation
LF	Late Finish Time
LS	Late Start Time
MRD	Maximum Daily Resource Demand
NGD	Node Gender Decision Method
NPVP	Net Present Value Problem
NP-HARD	Non-deterministic Polynomial Time Hard
OVERLOAD	Overload Resource Metric
PACK	Packing Method
PRED	Predecessor
PSO	Particle Swarm Optimization
PSPLIB	Project Scheduling Problem Library
RACP	Resource Availability Cost Problem
RAM	Random Access Memory
RCPSP	Resource Constrained Project Scheduling Problem
RCPSPDC	Resource Constrained Project Scheduling Problem with Discounted Cash Flows

RES	Resource
RID	Resource Idle Days
RLP	Resource Leveling Problem
SA	Simulated Annealing
SD	Start Date
SSQR	Sum of Squares Metric
SUCC	Successor
TCTP	Time-Cost Trade-off Problem
TF	Total Float
TPS	Temporal Scheduling Method
ub	upper bound

CHAPTER 1

INTRODUCTION

Planning and management are vital processes for projects to achieve desired goals at least within budgeted availabilities and with expected quality outcomes. The significance of planning and management is mostly seen in construction projects, where the environmental conditions affecting the process are highly changing. The function of good planning and management is to adapt the process to the dynamic conditions so as to complete the construction project within its budgeted time and cost availabilities and with the expected quality outcomes. Construction projects differ from other projects from the aspect of being unique endeavors that have a start time, finish time and a processing period in which all tasks are performed to complete the project. Tasks to complete a construction project may be similar to the tasks to complete another. However, mostly the affecting conditions are not the same and so each project requires its own project plan. Each project depends on a well prepared planning and management accordingly to be finished within its budgeted time, cost availabilities, and with the expected quality outcome. Most of the problems like delays, exceeding resource limits, exceeding budgeted cost, decreasing quality are all due to the lack of efficient planning and management.

There are several challenges faced while planning a construction project. Among those challenges can be listed the resource constrained project scheduling problem (RCPSP), the resource leveling problem (RLP), the time-cost trade-off problem (TCTP), the net present value problem (NPVP), and the resource availability cost problem (RACP). Resource constrained project scheduling problem focuses on performing the work with limited resources to finish the project in duration as short as possible. Resource leveling problem concentrates on optimizing the resource utilization of a project to minimize resource related cost issues and obtain the most effective resource schedule to complete the project in more economical way. Time-cost trade-off problem deals with the optimization of the time required to perform the works and the cost of these works. It tries to find the shortest duration to finish a project while at the same time keeps the cost as minimum as possible. The net present value problem aims to maximize the net present value where cash inflows and periodic expenses are considered. Finally, resource availability cost problem deals with different resource levels and corresponding cost values to find the best combination for a project. Throughout this thesis, the resource leveling problem (RLP) will be focused on only and studies carried out on this subject will be presented.

Construction projects are generally large projects that require a considerable amount of work carried out by large number of resources where resources may be labor and non-labor. These

resources may not be available always at desired quantities, or occupying high number of resources at the same time for short periods may become much costly than using them in less quantities but constant and spread to longer periods. Due to such financial issues related to resources, it is highly desirable to have a smooth and leveled resource utilization profile in which peak demands are lowered and fluctuations, causing unproductive working periods, are minimized. This can be accomplished by scheduling the activities such that their resource usages are summed up to form targeted resource utilization profile. Targeted resource utilization profile may vary depending on the type of the project. Yet for construction projects, generally, a bell shaped profile which has low peak demand is the most desirable and acceptable (Mutlu 2010).

The mostly used scheduling method in construction projects is the critical path method (CPM). In this method, the activities are scheduled to their earliest start times that satisfy the precedence relationships in between. Additionally, latest start times of activities are calculated and total float of each activity is determined. Some activities have zero total float which means they are not allowed to be started later than their earliest start times and they are critical. Any delay caused by these activities causes a delay in the project finish time. Other activities have positive total floats meaning that these activities have the chance to be started later than their earliest start time without causing any delay at the end of the project. Starting the activities on their earliest start times causes the most resources to be used at the beginning of the project. The corresponding resource usage profile usually comes out to have a high peak occurring at the early times of the project and fluctuations throughout the project span. The peak should not necessarily be seen at the beginning, but it may occur towards the end of the project where all the activities are scheduled to their latest start times, for example. Both of these cases are not wanted because of extra cost of irregular resource usage. High peak resource demands and fluctuations are seen as waste of time and cost due to inefficient planning. Therefore, many studies focused on leveling the resource utilization profile to reduce the high peak and to get a flatter resource usage and hence, to minimize resource related financing issues.

Leveling resources is done by changing start dates of non-critical activities (activities having positive total floats) to obtain schedules that have smooth resource usage profiles. To find the best schedule that gives the most desired resource usage profile, roughly all possible schedules should be generated and compared. This means that the effort required to find the best solution to RLP depends on the size of search of the overall solution space. RLP in nature is a non-deterministic polynomial time hard (NP-hard) problem (Neumann et al, 2003) and to find the exact solution special techniques should be applied.

Because of the nature of the NP-hard problems, solving to find the best solution requires immense computation time and memory as the solution space is very large and today's technology is still insufficient to handle it. Exact solution methods (like branch and bound method) can be useful for problems with few activities. But for problems with many activities, i.e. like in the case of most construction projects that have hundreds to thousands activities, finding the exact solution is almost impossible with current technology. Thus, the immediate growing idea has become to develop heuristic methods and solve for an

acceptable result in short time and low memory requirement. As a result of that, most studies have proposed heuristic methods to solve RLP. Many heuristic and metaheuristic methods have been developed and large problem instances have been solved for RLP in short time. From this aspect, heuristic and metaheuristic methods comprise an easy and useful way of solving large problems and generating resource leveled schedules. Nevertheless, the main issue with heuristic and metaheuristic methods is that it is not known how good of a solution they find. Since exact results of test problems are not available without exact solution methods, the results obtained by heuristic and metaheuristic methods cannot be compared to identify their performance and efficiency. Exact methods are needed to find the exact solutions of at least problem test sets and use the exact result of these problem sets as reference for performance comparison of heuristic and metaheuristic methods. Available studies in the literature use dynamic programming, implicit enumeration, branch and bound algorithm and integer linear programming methods for exact solution of RLP (Petrovic 1969, Mason and Moodie 1971, Easa 1989, Bandelloni et al. 1994, Younis and Saad 1996, Mattila and Abraham 1998, Neumann and Zimmermann 2000, Son and Mattila 2004, Mutlu 2010, Gather at al. 2010, Rieck et al. 2012). However, still the available methods are limited. In this thesis, an efficient branch and bound algorithm is proposed for the resource leveling problem. The aim behind the selection of this method is that its capability to be applied to solve for any objective function. Many techniques are implemented to improve the algorithm and increase its capability to solve larger problems. As a result, a branch and bound algorithm for the resource leveling problem has been developed with the implementation of many improvements and it has been tested through extensive computational experiments using four different objective functions.

The following assumptions are made for the RLP:

- Precedence relations, activity durations, activity resource requirements are predefined and all of them remain constant throughout the solution procedure.
- The complete project duration is fixed, i.e. the project finish time cannot be delayed.
- Activities remain continuous throughout their work span. Once they are active, they remain active until they are completed.
- Activities, utilize a uniform resource profile throughout their work span.

The proposed algorithm is constructed using C++ programming language. The problem sets used to test the developed algorithm are instances from the test set of Mutlu (2010), PSPLIB of Kolisch et al. (1997), benchmark instances (*rlp_j10, rlp_j20*) of Kolisch et al. (1999), problem instances (*ubo10, ubo20*) of Franck et al. (2001), and test set T_2 of Rieck et al. (2012). Experimented problem instances have 10 to 30 activities and include single resource and multi resource cases (1, 3 and 5 resource types). All problem sets and computational results are presented in Chapter 7.

The remainder of this thesis is organized as follows. In Chapter 2, the literature review is provided. In Chapter 3, the objective functions used for RLP are stated. In Chapters 4 and 5,

the developed branch and bound algorithm is presented in detail. In Chapter 6, a mixed-integer linear model is proposed for the resource leveling problem. In Chapter7, the computational test results are given, and finally in Chapter 8, the conclusion of the study and outlook for further research is outlined.

CHAPTER 2

LITERATURE REVIEW

There are limited studies focusing on resource leveling problem. Besides, comparing the literature of heuristic and metaheuristic methods with exact methods of RLP, it is seen that there are less studies on exact methods since their nature is more complex and they require relatively high effort to construct their algorithms. Actually, researchers tend to concentrate on heuristic and metaheuristic methods as they are generally simpler than exact methods and they are capable of finding near optimal results to large problems with low effort, which is most of the time acceptable and very useful. However, as stated before, the need for exact solutions is inevitable as to be used for performance analysis of heuristic and metaheuristic methods. In this chapter, a literature review is given on heuristic and metaheuristic methods in simple way to generate the overview of RLP solving strategies and the literature review on exact methods and the studies that are inspired from are stated to give the reference of the main study of the thesis.

2.1 Literature Review of Heuristic and Metaheuristic Methods for Solving RLP

Heuristic methods are usually simple methods developed for an optimization problem. They are problem solving strategies to create a good solution with step by step construction or to improve a present solution with several trials of changing problem parameters. Burgess and Killebrew (1962), proposed one of the first heuristic methods to solve RLP. This method simply changes the start dates of the non-critical activities in some ordered steps for a number of cycles until to find the best result. On the other hand, metaheuristic algorithms are problem type independent methods and can be implemented to variety of optimization problems. They are more complex methods but usually are stronger in solving more difficult problems. Examples for metaheuristic methods are genetic algorithm, particle swarm optimization, ant colony, simulated annealing etc.

The heuristic method that Burgess and Killebrew (1962) have proposed is a priority based and simple shifting method. After Burgess and Killebrew (1962), most of studies also have focused on priority based shifting techniques (Galbreath 1965, Woodworth and Willie 1975, Wiest and Levy 1977, Harris 1978, Martinez and Ioannou 1993). Among these studies, a multi-project multi-resource case for RLP has been also studied (Woodworth and Willie, 1975). Harris (1990) has introduced a new method called Packing Method (PACK) to solve RLP by minimizing the moment of resource profile. Martinez and Ioannou (1993) and Hiyassat (2000) also have studied PACK method by modifying the minimization of moment of the resource profile. Hiyassat (2000) has considered activity resource requirements and

free floats for the selection of activities to be shifted. In Hiyassat (2001), minimum resource moment has showed well results for projects with multiple resources by a modification in the method. Cristodoulou et al. (2010) have developed the minimum moment and PACK methods by allowing the activities to be stretched or compressed. They have stated that the resource rate of the activities can be adjusted to obtain better resource usage profiles while at the same time they have considered maximum daily resource demand and solved for RLP in this way. A method called "The entropy-maximization method" has been also proposed in the paper. The first study that uses a different method than priority based methods is based on using neural network (Savin et al. 1996, and Kartam and Tongthong 1998).

Genetic algorithms have become popular in the following years and most of the researches have suggested RLP solving strategies based on GA's (Chan et al. 1996, Leu and Yang 1999, Hegazy 1999, Son and Skibniewski 1999, Leu et al. 2000, Wang and Zheng 2001, Oral et al. 2003, Zheng et al. 2003, Senouci and Eldin 2004, Chen et al. 2005, Han and Sun 2006, Hwang and He 2006, Roca et al. 2008, Bettemir 2009, El-Rayes and Jun 2009, Doulabi et al. 2010, Jun and El-Rayes 2011, Ponz-Tienda et al. 2013). The natural evolution inspires genetic algorithms. GA generates a population of initial solutions. The solution data are coded to chromosomes and by crossing over or mutation of chromosomes new generations (chromosomes) are produced. GA gives the good solutions a chance of survival and passes their chromosomes to the next generations to improve the result further. From this aspect, GA has a very strong search capability in the solution space and can be applicable to various types of problems. Yet, still there are some weaknesses of GA. Hegazy (1999) has applied random activity priorities to enhance GA. Son and Skibniewski (1999) used simulated annealing (SA) to have improved searching capabilities by escaping from local optimal points. Leu et al. (2000) have introduced a decision support system (DSS) to increase the capabilities of genetic algorithm. Senouci and Eldin (2004) have developed a GA based model to solve for total project cost optimization problems considering the precedence relations and multiple crew strategies.

El-Rayes and Jun (2009) have introduced new resource utilization metrics and developed a GA based model to solve problems using different objective functions. They have solved a problem with 20 activities and single resource type. The new metrics that El-Rayes and Jun (2009) have introduced aim to reduce the number of release-rehire and idle resource times. The first metric states that during the transition from high demand period to low demand period resources may be released and again for a coming high demand period resources are rehired. High number of release-rehire periods affects the performance of the resources by disturbing the experiment gaining of workers and affecting working in a comprehensive manner. Thus, this metric focuses on reducing the number of release and rehire periods. The second metric is about when the resources are not released while passing from higher demand periods to lower demand periods but kept waiting for the coming high demand period. In this case waiting resources are idle and they increase the resource cost. This metric tries to minimize the resource idle periods. As a limiting factor, maximum daily resource demand can be integrated with the above-mentioned metrics to prevent high peak demands. It has been stated that the new metrics have high capability of optimizing resource usage profiles where resource fluctuations are minimized and a bell-shaped histogram is

obtained. The second metric that is about resource idle periods will be explained in detail in Chapter 3 as it has been used in the experimental work of this study. Doulabi et al. (2010) have used a repair mechanism in GA to improve converging capabilities of GA and reduce computation time. Roca et al. (2008) and Jun and El-Rayes (2011) have used GA to solve RLP and RCPSP simultaneously.

The most recent study on metaheuristic methods is by Ponz-Tienda et al. (2013) using genetic algorithm based methods for solving RLP. They have proposed an adaptive genetic algorithm using Weibull distribution to estimate global optimality and terminate the procedure. The developed algorithm has been tested using problem sets of 30, 60 and 120 activities size from Project Scheduling Problem Library, PSPLIB. The results have been compared with the early start schedules to analyze the performance of the algorithm.

Neumann and Zimmermann (1999) have developed a polynomial priority based metaheuristic algorithm. They tested their algorithm using different objective functions and analyzed the results using empirical performance analysis techniques. Ballestin et al. (2007) have presented a population based method of integrated greedy type for make-to-order production and then, they applied this method to solve RLP. One of the other most popular metaheuristic methods is particle swarm optimization (PSO). Qi et al. (2007), Pang et al. (2008), Guo et al. (2009) has studied PSO to solve RLP and some solved two examples in both PSO and GA and stated that PSO has better performance compared to GA. Other studies on metaheuristic methods by Xiong and Kuang (2006), and Geng et al. (2010) have used ant colony algorithm. Xiong and Kuang (2006) have integrated the serial programming generation scheme with ant colony algorithm.

Some of the literature studies have included small test samples in their experiments. Hegazy (1999) has used test samples of at most six resource types and 20 activities. Leu et al. (2000) has studied with problems of three resource types and 9 activities at most. Bettemir (2009) has compared five different GA algorithms using problems up to 13 activities.

Heuristic and metaheuristic problems propose fast and easy solution procedures for project scheduling problems. However, their solution capability is not known as they are not guaranteed to find the optimal solution but they find the best solution they can find. For the performance analysis, the need to know exact results is inevitable. The literature of heuristic and metaheuristic methods proposed for project scheduling problems especially those that have focused on RLP has been investigated. Therefore, an idea of the state of RLP solving approaches can be generated. Summary of the heuristic and metaheuristic literature is given in Table 2.1 as grouped according to the project scheduling problems that have been focused on and the methods that have been employed. Studies are sequenced according to their publication year within the groups and some general remarks are stated.

Table 2.1 Literature review of heuristic and metaheuristic methods for RLP and other project scheduling problems (1/3)

Year	Authors	Developed Methods	Scheduling Problem
1962 **1965** **1975** **1977** **1978**	Burgess and Killebrew Galbreath Woodworth and Willie Wiest and Levy Harris	Priority Based Simple Shifting	Project Scheduling Problems
Notes:	Simple shifting heuristics or priority-rule based methods for project scheduling problems subject to precedence constraints.		
1990 **1993** **2000** **2001** **2010**	Harris Martinez and Ioannou Hiyassat Hiyassat Cristodoulou, Ellinas and Kamenou	PACK Method with Priority Based Simple Shifting	RLP
Notes:	Minimum moment method, modified minimum moment method, has been utilized based on PACK method. Activity selection criteria have been defined for some studies. Activity stretching is allowed and entropy-maximization method has been introduced in the study of Cristodoulou et al. (2010)		
1996 **1998**	Savin, Alkas and Fazio Kartam and Tongthong	Neural Network	RLP
Notes:	Neural network based methods have been applied for resource leveling problem.		
1996 **1999** **2004** **2008** **2011**	Chan, Chua and Kannan Hegazy Senouci and Eldin Roca, Pugnaghi and Libert Jun and El-Rayes	GA Based	RLP, RCPSP
Notes:	GA based procedures have been introduced to minimize deviations from the available resource limits. Random activity selection has been defined and double moment method has been employed to the GA module by Hegazy (1999). Minimization of total project cost has been studied (Senouci et al., 2004). Resource leveling and resource allocation problems have been considered simultaneously (Roca et al., 2008, Jun et al., 2011).		

Table 2.1 Literature review of heuristic and metaheuristic methods for RLP and other project scheduling problems (2/3)

Year	Authors	Developed Methods	Scheduling Problem
1999	Son and Skibniewski	SA Based	RLP
Notes:	Sum of squares method is used as for objective function.		
2000	Leu, Yang and Huang	GA Based	RLP
Notes:	A Decision Support System (DSS) has been introduced to the GA based procedure. Minimization of absolute deviation from average resource level has been aimed.		
2003	Oral, Oral, Bozkurt and Erdis	GA Based	RLP
Notes:	Minimum deviations from the average resource demand has been utilized to level resources by a GA based algorithm.		
2003	Zheng, Ng and Kumaraswamy	GA Based	RLP
Notes:	Minimum moment method has been utilized to level resources. Weighted multiple resource cases has been discussed to represent effects of different resource types.		
2009	El-Rayes and Jun	GA Based	RLP
Notes:	Two new metrics of Release-Rehire and Resource Idle Days have been defined, GA based module has been developed to solve for RLP using these new metrics.		
2010	Doulabi, Seifi and Shariat	GA Based	RLP
Notes:	Local search heuristic and repair mechanism have been integrated with the genetic algorithm to solve RLP.		
2013	Ponz-Tienda, Yepes, Pollicer and Moreno Flores	GA Based	RLP
Notes:	An adaptive GA with utilization of Weibull distribution for decision of global optimality has been developed. Test sets from PSPLIB have been solved to test the algorithm.		

Table 2.1 Literature review of heuristic and metaheuristic methods for RLP and other project scheduling problems (3/3)

Year	Authors	Developed Methods	Scheduling Problem
1999 **2009**	Leu and Yang Bettemir (PhD. Thesis)	GA Based	RLP, RCPSP, TCTP
Notes:	Generic GA based algorithms have been developed to solve general project scheduling problems. Five different GA based metaheuristic algorithms have been developed and tested by Bettemir, 2009.		
1999	Neumann and Zimmermann	Polynomial priority based heuristic	RLP, RCPSP
Notes:	A polynomial priority based metaheuristic method has been developed to solve RLP using several different objective functions.		
2007	Ballestin, Schwindt and Zimmermann	Population based greedy heuristic	RLP
Notes:	Optimization on make-to-order production has been offered and then this model has been adapted to resource leveling problem using population based greedy method.		
2007 **2008** **2009**	Qi, Wang and Guo Pang, Shi and You Guo, Li and Ye	PSO	RLP
Notes:	Particle swarm optimization based metaheuristic algorithm has been developed to solve resource leveling problem. A constriction factor has been defined to prevent early convergence to a local optimal in Pang et al. (2008).		
2006 **2010**	Xiong and Kuang Geng, Weng and Li	Ant Colony	RLP
Notes:	Hybrid model of serial programming generation scheme with ant colony (Xiong et al., 2006) and an oriented ant colony (Geng et al., 2010) models have been proposed to solve RLP.		

2.2 Literature Review of Exact Methods for Solving RLP and other Project Scheduling Problems

Studies proposing exact solution methods for the RLP in literature are limited. Therefore, exact methods that are used for RLP, RCPSP and combination of cost and these problems studied in literature have been presented in this section. Main exact solving algorithms are based on dynamic programming, implicit enumeration, branch and bound algorithm and linear integer programming methods.

Agin (1966) has presented a general description of branch and bound algorithms. It has demonstrated that branch and bound methods are widely applicable on combinatorial problems with non-linear, discontinuous or non-mathematically defined objective functions under variety of constraints. Basic characteristics of branch and bound procedure have been explained and points that affect the efficiency of the method have been stated.

As a first contribution on exact methods of resource leveling problem, Petrovic (1969) offered a multistage dynamic programming approach. One of other first studies in the literature that deals with exact methods is by Mason and Moodie (1971). This study has focused on minimization of cost and resource utilization combined together. Elongation of project duration is allowed, and any delay on project finish date is penalized appearing in the cost function. Similarly, resources exceeding available resource limits are also penalized and seen in the cost function. The solution approach has been based on branch and bound algorithm. During scheduling an activity possible scenarios have been regarded and a lower bound has been calculated for each possibility. The possibilities to be kept have been decided according to the lower bound. Schedules that have over allocated resources have been also eliminated. The proposed algorithm has been tested using a 25 activity network. Results and parameters affecting computation time have been stated in the paper (Mason and Moodie, 1971). Ahuja (1976) has studied to level resource fluctuations by using enumeration method.

Patterson (1984) has studied the RCPSP problem using three exact methods and compared their performances. These three methods have been branch and bound, bounded enumeration and implicit enumeration. Based on the test results, branch and bound method has been reported to solve all instances out of 110 problems in 5 minutes. Bounded enumeration has recorded for some instances the shortest computation time. Implicit enumeration has required the least computation storage. Patterson (1984) has stated that branch and bound algorithm has more performance as it directs the search in the solution space to more promising regions.

Easa (1989) proposed a linear integer programming method for RLP that minimizes the deviations from average resource demand. Small to medium sized problem instances have been solved for optimality and optimal resource profiles have been compared with the early schedule resource profiles. In the paper, it has been stated that because high number of variables and constraints have been needed to define the model, the implementation of the linear integer model becomes difficult to apply it for practical purposes. Karshenas and Haber (1990) also have developed a linear integer programming based method aiming to

minimize the sum of cost of resources and the project duration. Two simple examples have been solved and it has been reported that the duration of the solution schedule has been optimal and the resource usage cost has been low.

Demeulemeester and Herroelen (1992) have proposed a method based on branch and bound algorithm having a depth-first methodology for RCPSP. At an instant, where feasible partial schedules are constructed from an eligible activity that is to be scheduled yet, nodes are generated and the ones having high lower bound values according to the bounding strategy are fathomed. 110 test problem set of Patterson (1984) has been solved at an average computation time of 0.215 seconds. It has been reported that the offered method has been faster than the developed methods earlier and requires low storage memory during its computation to solve for RCPSP. Afterwards, Shah et al. (1993) have studied a linear integer model to find the minimum resources to complete a project. Bandelloni et al. (1994) have offered a non-serial dynamic programming model that reduces the absolute deviations from a targeted resource level.

Following these studies Demeulemeester (1995) suggested to use branch and bound algorithm for solving resource availability cost problem. A small bridge project was used in addition to the problem set of Patterson (1984) to test the proposed algorithm. The effect of additional resource types on computation time was also observed. It has been reported that the computational time increases as additional resource types are included since the search space of the efficient solutions expands. Younis and Saad (1996) occupied a mathematical model to optimize resource utilization of a project, Icmeli and Erenguc (1996) used a depth-first branch and bound algorithm to solve resource constrained project scheduling problem with discounted cash flows (RCPSPDC). Demeulemeester and Herroelen (1997) developed the depth-first branch and bound algorithm to solve the generalized RCPSP. In this algorithm precedence based lower bound calculations have been added to several dominance rules in order to fathom nodes that are non-promising. The same problem set of Patterson (1984) were used to test the algorithm. 109 instances out of 110 problems have been solved in an average computation time of 8.1065 seconds.

Mattila and Abraham (1998) focused on resource leveling problem of linear scheduling projects like highway construction projects, high buildings, pipeline construction projects etc. Linear integer programming has been adapted for this problem and the absolute deviation from the average resource demand has been minimized. The programming was carried out using a software program called LINDO, and the obtained leveled resource utilization profile for a highway construction project has been reported. As most of the researchers has stated, Mattila and Abraham (1998) also has mentioned on the hardness of implementation of large size problems due to the high number of variables and constraints required to define the problem.

Another study using branch and bound algorithm for RCPSP is by Brucker et al. (1998). In this study, a tabu search algorithm has been integrated to the branch and bound algorithm in the root of the search tree. This way the tree formation procedure has been initialized with a better schedule. In addition a linear integer based lower bound calculation has been adapted

to the procedure. Brucker et al. (1998) have experimented problem sets with 30 and 60 activities having 4 types of resources. 326 of 480 problem instances of 60 activities have been solved for optimality within one hour. De Reyck and Herroelen (1998) also presented a branch and bound algorithm with depth-first methodology for RCPSP. Minimal delaying alternatives method has been employed to overcome resource infeasible nodes. Extensive experimentation results have been presented and problem networks up to 100 activities have been solved for optimal solution.

One of the most extensive papers has been published by Neumann and Zimmermann (2000) concentrating on heuristic and exact procedures to solve RLP and net present value problem. In this paper minimization of resource costs, minimization of deviations from a given resource level and minimization of deviations of consecutive time periods are used as objective functions to solve RLP with and without resource limitations. For the stated exact procedure branch and bound and truncated branch and bound algorithms have been utilized. The branch and bound algorithm has been based on enumeration of feasible start times. Nodes represent partial schedules and leaves (final nodes that do not have any remaining unscheduled activity) represent complete feasible schedules. Selection of the node to be branched from is done according to its lower bound value. The node having the smallest lower bound among newly formed nodes is branched. Branching new nodes from the selected node is carried out by scheduling the activity that has the lowest total float among remaining eligible activities. In truncated branch and bound algorithm a heuristic is used to cut non-promising branches and allow only most promising branches to grow up the solution. Moreover, a tabu search procedure has been integrated to the algorithms mentioned above. Neumann and Zimmermann (2000) have employed an extensive problem set for experiments. Instances have activities from 10 to 500 and resource types from 1 to 5. Most of the problems in the sets that consist of up to 20 activities have been solved for optimality within 100 seconds. It has been declared that, instances with 20 activities and 5 resources have been solved for optimal solutions for the first time in the literature.

Nübel (2001) has developed a depth-first branch and bound algorithm to solve for resource renting problem subject to temporal constraints. The aim of resource renting problem is to minimize resource availability costs. Time-independent and time-dependent renting costs have been included in the study. The solution approach has been based on enumeration of a finite set of schedules that is proven to contain the exact solution. A computational study has been devised over a randomly generated test set and results are reported.

Vanhoucke et al. (2001) have suggested a branch and bound algorithm for maximizing the net present value. New upper bound computation strategies and more branching strategies have been employed to reduce the search tree size in considerable amounts. Problem sets of Patterson (1984) and Icmeli and Erenguc (1996) have been used for testing the algorithm. It has been stated that instances up to 30 activities and 4 resource types have been solved optimally for net present value problem.

Zamani (2001) has introduced a different approach to the selection of node for branching in solving RCPSP. In this paper, the node with the minimum lower bound is branched, but

different than Neumann and Zimmermann (2000) the node with smallest lower bound among all open nodes is selected instead of taking into account only newly formed nodes. Zamani (2001) has constructed the algorithm to follow the minimum lower bound nodes for branching process until the optimal schedule is found. It has been stated that branching from a node with the minimum lower bound does not require large memory to keep open nodes and does not require large comparability time to trace in the tree and select the node to branch from. The developed algorithm has been tested using a test problem set that includes networks up to 100 activities and 6 different resource types. Experimentation results have been reported in the paper.

Son and Mattila (2004) have approached resource leveling problem from a different manner. Up to that time, researchers assumed that activities cannot be split, i.e. once they are started they should remain continuous till it is finished. In this paper, a linear program of binary variable model has been introduced to level resource usage of a project where activity splitting is allowed. Two example projects have been solved and it has been stated that the resulted resource profiles where activities can be split are more representing actual construction projects.

The study of Jiang and Shi (2005) is another study focusing on solving RCPSP. In that paper, enumerative branch and bound procedure has been employed. In this procedure, similar to truncated branch and bound procedure of Neumann and Zimmermann (2000), a cutting rule has been applied to eliminate bad schedules. 110 problem instances set of Patterson (1984) could be solved in reasonable computational time. It has been also stated in this paper that computational time is not a big deal for solving project scheduling problems as scheduling in a real life is not repeated every time throughout a project life.

Mutlu (2010) has studied the resource leveling problem by developing an algorithm based on depth-first strategy branch and bound procedure in his master thesis. Improved lower bound calculation strategies have been presented to reduce the stored feasible partial schedule start time packages. The developed algorithm have been stated to solve instances up to 20 activities and 4 resource types using four different objective functions. In addition to traditional sum of squares of daily resource usages and absolute deviations from a targeted resource level, minimization of resource idle days (El-Rayes and Jun, 2009) and minimization of weighted combination of resource idle days and maximum daily resource demand have been employed for testing. A test set consisting of small-scale problems has been solved for RLP, and the objective function of minimization of resource idle days has been used for testing for the first time.

One of the most recent studies has been published by Gather et al. (2010). This paper concentrates on solving resource leveling problem subject to general temporal constraints. It has been mentioned that the resource leveling problem even with single resource is an NP-hard problem (Neumann et al., 2003). To solve NP-hard resource leveling problem they have introduced a new enumeration scheme. They have proposed an appropriate solution procedure combining the enumeration scheme with the branch and bound procedure to solve for RLP. For this purpose, a tree-based enumeration scheme has been developed where a

concept of Gabow and Myers (1978) has been integrated to overcome the significant weaknesses of the tree-based approach of Nübel (2001). The proposed algorithm has been tested by a comprehensive computational study using the well-known *rlp_j10* and *rlp_j20* test sets of Kolisch et al. (Benchmark Instances for Project Scheduling Problems, 1999). It has been declared that the proposed algorithm outperformed the methods known in the literature. Large number of instances with 20 activities of test sets of Kolisch et al. (1999) have been solved for optimality for the first time. It has been pointed out that mixed integer linear programming can be suitable to model resource leveling problem appropriately for the future studies.

Following the study of Gather et al. (2010), Rieck et al. (2012) published a paper that has introduced a mixed-integer linear programming technique for the resource leveling problem subject to general temporal constraints. In this study, sum of squares of the resource utilization cost and the "overload problem" have been considered. The overload problem tries to minimize the cost due to exceeding a resource limit. Generally, this limit is taken to be the average resource demand. The overload problem is very similar to the minimization of absolute deviations from the average resource demand from the aspect of their functions on leveling. This objective function will be explained in detail in Chapter 3 as it has been employed in the experimentation of the thesis study. Rieck et al. (2012) have presented new mixed-integer linear model formulations and domain-reducing preprocessing techniques. Lower and upper bounds for resource requirements at particular points in time, effecting cutting planes and problem-specific preprocessing techniques have been implemented to enhance the models. The developed algorithm has been constructed using CPLEX 12.1. The most generic experimentation up to that time has been carried out in this study. Medium-sized problems and problem sets of Kolisch et al. (1999) have been solved. Instances with 30 activities and up to 5 resources types have all been solved for optimality. Moreover, some instances up to 50 activities have been solved for the first time in the literature. Tests have been performed for the strict project completion duration case and for extended project completion duration. Test results of Rieck et al. (2012) have been used for performance comparison of the algorithm proposed in this thesis study.

Looking up to the literature it is seen that only few studies have focused on optimally solving resource leveling problem. As studies of other project scheduling problems can be inspired from and the applied techniques could be adapted to any other specific scheduling problem, publications related to branch and bound procedures and other utilized techniques have been stated. The summary of the literature review for exact solutions of RLP and other project scheduling problems have been presented in tabular form in Table 2.2 sequenced in chronological order. General remarks related to each study have also been stated.

Table 2.2 Literature review of exact solutions for RLP and other project scheduling problems (1/4)

Year	Authors	Developed Methods	Scheduling Problem
1969	Petrovic	Dynamic Programming	RLP
Notes:	Multistage dynamic programming approach has been presented as an early study for solving RLP.		
1971	Mason and Moodie	Branch and Bound	RLP and Project Duration Minimization
Notes:	Project duration extension is allowed but delays and over allocated resources are penalized using a cost function. The importance of lower bound calculations for algorithm performance has been stated.		
1984	Patterson	Bounded Enumeration, Branch and bound, Implicit Enumeration	RCPSP
Notes:	Using special rules, non-promising solution space regions are eliminated. A test set of 110 instances has been used for performance comparison of the proposed algorithms.		
1989	Easa	Linear Integer Model	RLP
Notes:	A linear integer optimization model has been developed to solve resource leveling problem. Deviations from average resource demand and resource deviations of consecutive time periods have been aimed to be minimized.		
1990	Karshenas and Haber	Linear Integer Model	Resource Cost Minimization
Notes:	A general approach to optimize project scheduling problems. Financially leveled resource histograms as well as optimal project duration have been obtained by the proposed method.		
1992	Demeulemeester and Herroelen	Branch and Bound	RCPSP
Notes:	A depth-first branch and bound algorithm has been developed. Partial schedules that have bad lower bounds are fathomed. 110 test problem of Patterson (1984) has been solved in short durations.		

Table 2.2 Literature review of exact solutions for RLP and other project scheduling problems (2/4)

Year	Authors	Developed Methods	Scheduling Problem
1993	Shah, Farid and Baugh	Linear Integer Model	Minimum Resource Requirement
Notes:	Minimum resources to complete a project within time has been investigated.		
1994	Bandelloni, Tucci and Rinaldi	Non-Serial Dynamic Programming	RLP
Notes:	Minimization of absolute deviations from a targeted resource level has been used as objective function for resource leveling problem.		
1995	Demeulemeester	Branch and Bound	Resource Availability Cost Problem (RACP)
Notes:	Branch and bound method for cost minimization of available resources levels has been proposed. It has been stated that additional resources increases the complexity of problem.		
1996	Icmeli and Erenguc	Branch and Bound	RCPSP
Notes:	Depth-first methodology has been occupied for the branch and bound algorithm to solve RCPSP in which discounted cash flows take place and net present value has been maximized.		
1996	Younis and Saad	Mathematical Model	RLP
Notes:	Multi resource projects have been mathematically modeled to optimize their resource utilization profiles.		
1997	Demeulemeester and Herroelen	Branch and Bound	RCPSP
Notes:	Extensive version of depth-first branch and bound algorithm suggested by Demeulemeester et al., (2002) has been presented. Extensive experimentation has been carried out.		
1998	Brucker, Knust, Schoo and Thiele	Branch and Bound	RCPSP
Notes:	Tabu search has been integrated to the algorithm to start the search tree from a good point. Linear integer programming based lower bound calculation has been presented.		

Table 2.2 Literature review of exact solutions for RLP and other project scheduling problems (3/4)

Year	Authors	Developed Methods	Scheduling Problem
1998	De Reyck and Herroelen	Branch and Bound	RCPSP
Notes:	Depth-first methodology of branch and bound has been integrated with dominance rules. Extensive testing has been performed and instances up to 100 activities have been solved.		
1998	Mattila and Abraham	Linear Integer Model	RLP
Notes:	Using linear integer model programmed in LINDO, resource optimization has been done for linear schedules. A highway project has been utilized for testing the algorithm.		
2000	Neumann and Zimmermann	Branch and Bound	RLP, Net Present Value Problem (NPVP)
Notes:	Different heuristic and exact methods are proposed to solve for RLP and NPVP. Instances up to 20 activities and 5 resources have been solved for the first time for RLP.		
2001	Nübel	Branch and Bound	Resource Availability Cost Problem (RACP)
Notes:	Depth-first branch and bound procedure has been proposed to solve for resource renting problem subject to temporal constraints. Computational test results of randomly generated problems have been reported.		
2001	Vanhoucke, Demeulemeester and Herroelen	Branch and Bound	Net Present Value Problem (NPVP)
Notes:	Net present value problem has been studied. Instances up to 30 activities and 4 resource types have been solved for optimality.		
2001	Zamani	Branch and Bound	RCPSP
Notes:	Selection of node having minimum lower bound value for branching process has been stated to reduce computation and storage memory to find the optimal solution.		

Table 2.2 Literature review of exact solutions for RLP and other project scheduling problems (4/4)

Year	Authors	Developed Methods	Scheduling Problem
2004	Son and Mattila	Linear Integer Model	RLP
Notes:	Activity splitting has been allowed for the resource leveling problem. This way more realistic project representations have been stated to be obtained.		
2005	Jiang and Shi	Branch and Bound	RCPSP
Notes:	Enumerative branch and cut procedure have been employed. 110 instances set of Patterson (1984) have been reported to be solved for optimality.		
2010	Mutlu (Master Thesis)	Branch and Bound	RLP
Notes:	Depth-first based approach has been developed for RLP. Lower bound calculation strategy has been improved. Small scale test instances have been solved for the minimization of resource idle days and other traditional objective functions.		
2010	Gather, Zimmermann and Bartels	Branch and Bound	RLP
Notes:	A new enumeration scheme has been integrated to the branch and bound algorithm. Test sets *rlp_j10* and *rlp_j20* of Kolisch et al. (1999) and *ubo10* and *ubo20* of Frank et al. (2001) have been solved and results have been reported.		
2012	Rieck, Zimmermann and Gather	Mixed-Integer Linear Programming	RLP
Notes:	New methods based on mixed-integer linear modeling have been proposed to solve RLP using sum of squares metric and overloading metric. Extensive computational tests have been carried out and instances up to 50 activities and 5 resources have been solved to optimality for the first time.		

CHAPTER 3

OBJECTIVE FUNCTIONS USED IN RLP ALGORITHMS

There are several objective functions used for the resource leveling problem. Most of the objective functions force the solution procedure to yield a flat resource utilization histogram where fluctuations are minimized. However, this rectangular shape of histogram may not be possible to obtain most of the times, and in actual construction projects a bell-shaped resource profile would be more practical. Four objective functions have been used for testing. They are minimization of weighted sum of square of daily resource usages, weighted sum of absolute deviations of daily resource usages from the targeted (generally average) resource level, weighted sum of daily overloaded resource amounts, and finally weighted sum of weighted combination of maximum daily resource demand and resource idle days. Resource weights can be considered as resource costs or else for which the pressure of each resource is emphasized. In the following sections these objective functions are stated in detail and corresponding formulations are given.

3.1 Sum of Squares of Daily Resource Utilization Method, (SSQR)

In this method the objective function minimizes the sum of squares of daily resource usages where summation is performed by weights for different resource types. This metric affects all daily resource usages under and over the average resource demand individually. It has a stronger capability of peak minimization than the absolute deviations from average resource demand and overloaded resource amount metrics. The formulation of SSQR metric has been given in Equation (3.1).

$$f_{SSQR} = \sum_{k=1}^{K} w_k \sum_{t=1}^{T} r_{kt}^2 \tag{3.1}$$

where; f_{SSQR} is the objective function to be minimized, K is the total number of resource types, k is the resource type, w_k is the weight of resource k, T is the total project duration, t is a day in the project span, r_{kt} is the resource usage of resource type k at the day of t.

In Figure 3.1 a sample resource utilization profile of 10 days length is given. The squares of daily resource usages are calculated and this sum has been aimed to be minimized by the SSQR objective function. As high resource usages make high contribution to the

summation, they are forced to approach average values in which the contribution of the total resources are minimum. In the right hand side profile of Figure 3.1, the possible best resource profile that has the lowest SSQR value has been shown. Note that the convergence point of SSQR yields a flat, rectangular-shaped resource utilization profile in the best case.

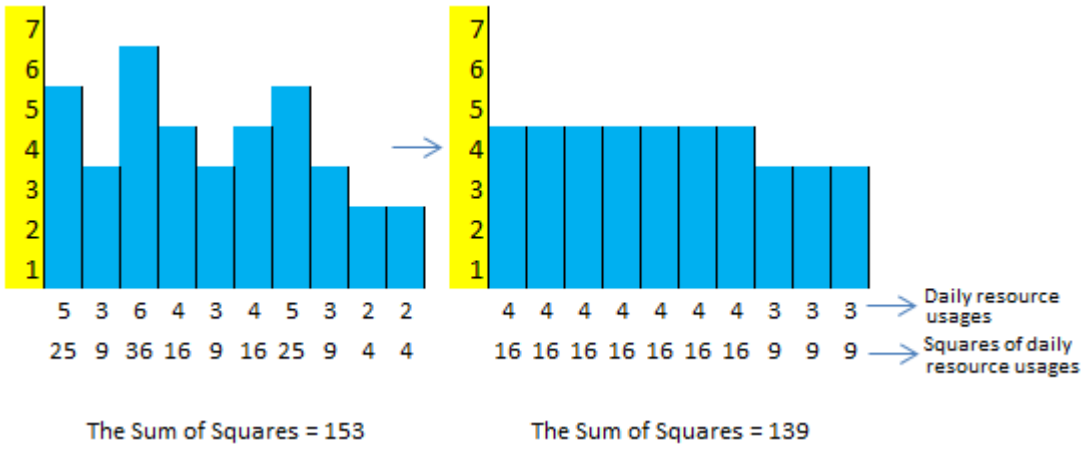


Figure 3.1 Sum of squares of daily resource usages of a sample profile

3.2 Absolute Deviations of Resource Utilization from the Targeted Resource Utilization Level Method, (ABSDEV)

This method calculates the absolute deviations of daily resource usages from the targeted resource utilization level. The targeted resource utilization level is generally taken as the average resource demand but it may take different values to achieve different resource levels. In the experimentation work of the thesis, the targeted resource level has been taken as the average resource demand calculated based on standard rounding. The formulation of this objective function has been given in Equation (3.2).

$$f_{ABSDEV} = \sum_{k=1}^{K} w_k \sum_{t=1}^{T} |r_{kt} - y_k| \tag{3.2}$$

$$y_k = \left\lfloor \left(\frac{1}{T} \sum_{t=1}^{T} r_{kt} \right) + 0.5 \right\rfloor \tag{3.3}$$

where; f_{ABSDEV} is the objective function to be minimized, K is the total number of resource types, k is the resource type, w_k is the weight of resource type k, T is the total project duration, t is a day in the project span, r_{kt} is the resource usage of resource type k at the day of t, and y_k is the average resource utilization level as calculated in Equation (3.3). The addition of 0.5 in the average calculation is to ensure standard rounding before flooring operation.

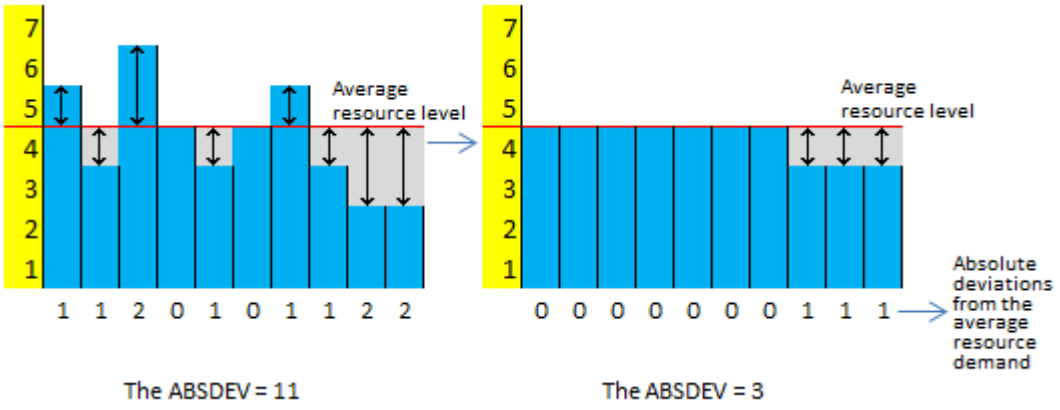


Figure 3.2 Absolute deviations of daily resource usages from the average resource demand of a sample profile

In Figure 3.2 the same resource utilization profile in section 3.1 is given. The absolute deviations of daily resource usages from the average resource demand are calculated and this sum has been aimed to be minimized by the ABSDEV objective function. All the resource usage bars are forced to approach the average level. In the right hand side profile of Figure 3.2, the possible best resource profile that has the lowest ABSDEV value has been shown. Note that the convergence point of ABSDEV yields a flat, rectangular-shaped resource utilization profile in the best case similar to that of SSQR.

3.3 Overloaded Resource Amounts Over the Targeted Resource Utilization Level Method, (OVERLOAD)

This metric is very similar to the ABSDEV metric from the aspect of leveling capability. The main difference is that in resource overloading metric the resources that exceed the targeted resource utilization level, i.e. the positive deviations, are taken into account. Considering the average resource demand is taken as the targeted resource level, in this objective function the aim is to minimize the overloading resources that exceeds the average resource demand (Rieck et al., 2012). In the experimentation of the thesis study, for some problem sets ABSDEV metric has been utilized, for some others OVERLOAD metric has been utilized to able to compare with previous testing. As their function on leveling is very

similar but the representation metrics are different only one of them has been employed for the experimentation of the same test set. The formulation of this objective function has been given in Equation (3.4).

$$f_{OVERLOAD} = \sum_{k=1}^{K} w_k \sum_{t=1}^{T} (overload_k) \tag{3.4}$$

$$where;\ \ if\ (r_{kt} - y_k) > 0 \rightarrow overload_k = (r_{kt} - y_k)$$
$$else\ \rightarrow overload_k = 0$$

and where; $f_{OVERLOAD}$ is the objective function to be minimized, K is the total number of resource types, k is the resource type, w_k is the weight of resource type k, T is the total project duration, t is a day in the project span, r_{kt} is the resource usage of resource type k at the day of t and y_k is the average resource utilization level as calculated in Equation (3.3).

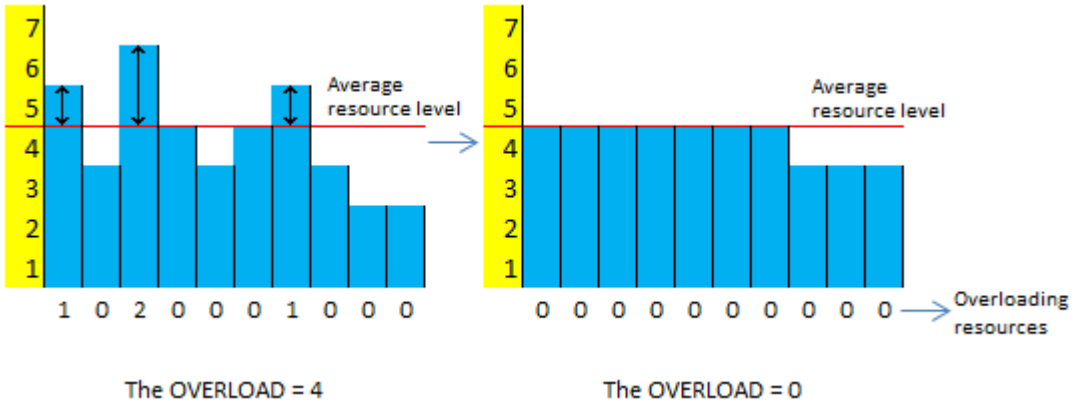
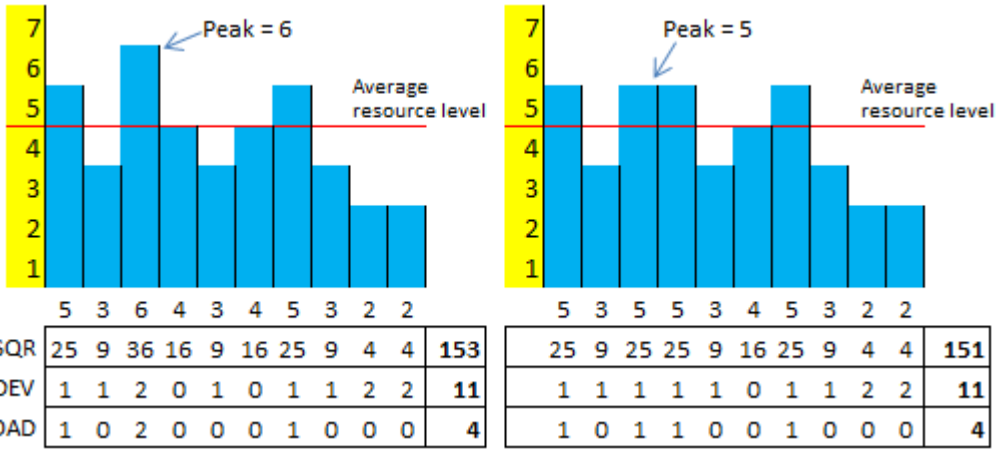


Figure 3.3 Overloading resources of a sample profile

In Figure 3.3 overloading resources are represented. Overloading resources are the exceeding amount of daily resource demands over average resource level. OVERLOAD metric tries to minimize those exceeding resource amounts. Therefore, all resource bars that are greater than the average resource level are forced to approach the average level. In the right hand side profile of Figure 3.3, the possible best resource profile that has the lowest OVERLOAD value has been shown. Again the convergence point of OVERLOAD yields a flat, rectangular-shaped profile in the best case similar to that of SSQR and ABSDEV.

As stated in section 3.1, ABSDEV and OVERLOAD metrics are not as good as SSQR metric in terms of minimizing the peak. To illustrate this difference the same sample resource profile in previous sections is used in Figure 3.4. The only change between the right side profile and the left side profile is that for the 3rd and 4th days resource demands are changed (6, 4 for the left side; 5, 5 for the right side for the 3rd and 4th days respectively). Right side profile has a lower peak of 5 while on left side profile the peak is 6. When the objective function values are considered, it can be seen that SSQR has a lower objective function value with the profile having lower peak (SSQR = 153 for the left side; SSQR = 151 for the right side) whereas other metrics have the same objective function values for both profiles. As a result, it can be said that SSQR metric obtains better resource utilization profiles with lower maximum daily resource demands. However, computation wise SSQR metric may be disadvantageous compared to other two metrics.



	5	3	6	4	3	4	5	3	2	2	
SSQR	25	9	36	16	9	16	25	9	4	4	**153**
ABSDEV	1	1	2	0	1	0	1	1	2	2	**11**
OVERLOAD	1	0	2	0	0	0	1	0	0	0	**4**

	5	3	5	5	3	4	5	3	2	2	
	25	9	25	25	9	16	25	9	4	4	**151**
	1	1	1	1	1	0	1	1	2	2	**11**
	1	0	1	1	0	0	1	0	0	0	**4**

Figure 3.4 Different capabilities of SSQR, ABSDEV and OVERLOAD metrics on peak minimization

3.4 Resource Idle Days and Maximum Daily Resource Demand Method, (RID-MRD)

The resource idle day (RID) is relatively a new metric offered by El-Rayes and Jun (2009). Any low demand period occurring between two high demand periods employ lower amount of resources unless high demand period resources are released. In that case, some of the resources are not employed and they are waited idle. The purpose with that metric is to eliminate such resource idle periods. However, RID itself does not take the care of peak demand. The convergence point of RID yields resource utilization profiles from rectangular-shapes to hill-shapes. Yet, as stated above, most of the times the resultant resource usage profiles have a high peak demand. The integration of maximum daily resource demand (MRD) metric is to ensure peak minimization and to guarantee the resultant profile be more like a bell shape rather than a hill shape. The weights for combination of RID and MRD

allow to define the pressure of which metric to be minimized more. For the rest of the experimentation of the thesis RID and MRD are equally weighted. The formulation of RID-MRD objective function has been given in Equation (3.5).

$$f_{RID-MRD} = f_{RID} w_{RID} + f_{MRD} w_{MRD} \tag{3.5}$$

$where;$

$$f_{RID} = \sum_{k=1}^{K} w_k \sum_{t=1}^{T} \left(min\left(max(r_{k1}, r_{k2}, \dots, r_{kt}), max(r_{kt}, r_{kt+1}, \dots, r_{kT}) \right) - r_{kt} \right) \tag{3.6}$$

$$f_{MRD} = \sum_{k=1}^{K} w_k * max(r_{k1}, r_{k2}, \dots, r_{kt}, \dots, r_{T-1}, r_{kT}) \tag{3.7}$$

$f_{RID\text{-}MRD}$ is the objective function to be minimized, K is the total number of resource types, k is the resource type, w_k is the weight of resource type k, T is the total project duration, t is a day in the project span, r_{kt} is the resource usage of resource type k at the day of t, w_{RID} is the weight of RID metric, w_{MRD} is the weight of MRD metric. w_{RID} and w_{MRD} are taken as 1 in this thesis.

In Figure 3.5 resource idle days and maximum daily resource demand (peak demand) are represented. As seen in the left hand side profile, for some periods resource demands are lower than the minimum of the maximums before and after. In these periods there are previously hired higher number of resources and for the coming periods there will be need of at least these resources again. Thus, the excess resources at that period are not released and they are idle. RID metric tries to minimize those idle resource days. Meanwhile, MRD tries to reduce the peak demand. In the right hand side profile of Figure 3.5, the possible best resource profile that has the lowest RID-MRD value has been shown. The convergence point of RID-MRD yields a bell-shaped resource utilization profile different than that of other objective functions.

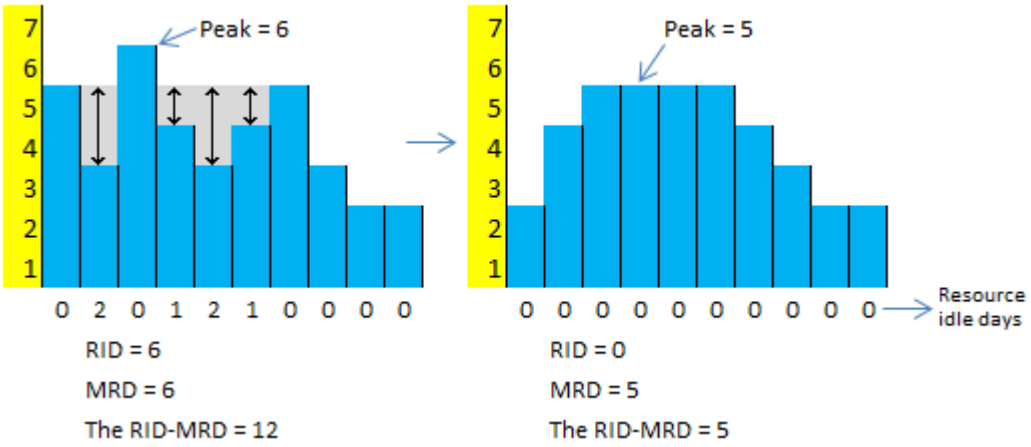


Figure 3.5 Resource idle days and maximum daily resource demand of a sample resource profile

CHAPTER 4

BRANCH AND BOUND ALGORITHM STRATEGIES FOR NP-HARD OPTIMIZATION PROBLEMS

Resource leveling problem is an NP-hard optimization problem in nature (Neumann et al., 2003). In NP-hard optimization problems, the solution space increases exponentially as the size of input increases. For example, in resource leveling problem, the size of the input highly depends on the number of the activities and the amount of floats that each activity has. As the number of activities increases, the size of the solution space increases exponentially. The proposed algorithm for the resource leveling problem in this thesis has been developed based on branch and bound algorithm. In order to improve its capabilities to solve large problems, this section explains the strategies implemented in the branch and bound algorithm. Most sensitive parts of the procedure have been addressed while applying new techniques so as to obtain best improvement results. In this chapter, a general overview of the branch and bound algorithm with basic definitions of branching strategy, lower bound calculation and upper bound calculation strategies, use of fathoming have been introduced. Then, storage memory and computation time dependency of branch and bound algorithm has been analyzed, and finally, propositions are made to focus on effective points for the improvement of the algorithm.

4.1 General Overview of Branch and Bound Algorithm with Basic Definitions

Branch and bound algorithm is a procedure where the permutations of an optimization problem are applied step by step by branching on the possibilities from each node till a complete solution is obtained while some possibilities are fathomed according to the bounding strategy during the solution tree formation. The solution procedure grows up a tree in which nodes represent partial solutions and the lowest level nodes, i.e. leaves, represent complete solutions to the problem. The starting point to the procedure is the root node. Starting from the root node, new nodes are generated via branching and some nodes are omitted by the bounding strategy. The branch and bound algorithm stages are described in the following sections.

Preparing Scheduling Information and Initial Data: Before starting the branch and bound procedure, the instance that is going to be solved is scheduled to find early and late start times for activities. Scheduling is performed with respect to either precedence relations or temporal constraints depending on the problem specifications. After early and late start times of activities are calculated, total floats for each activity are determined. The activities having zero total float are critical activities, and others are non-critical activities. Before

starting the tree formation procedure, earliest and latest start times, total float of activities are determined. The solution procedure will operate on non-critical activities using their available floats. The starting point to the tree formation is the root node. In the root node, only critical activities are started at their definite start times and all non-critical activities are not scheduled yet.

Lower Bound Calculation Strategy: In any node, resources of critical activities, scheduled non-critical activities and unavoidable periods are allocated. Then, the remaining resources are distributed to obtain the best resource histogram. The objective function value calculated for the obtained resource utilization profile specific to the issued node is called lower bound value of that node. In the root node, the resources of critical activities and unavoidable resources of non-critical activities are allocated. Remaining resources are distributed over the project span. The lower bound value calculated for the root node is the lowest value calculated throughout the solution procedure. As new nodes are generated, one of the non-critical activities is scheduled to a specific start time between its earliest and latest start times. The resources for this activity does not participate to the distribution over the allocated resources to obtain the best resource profile, but instead, allocated accordingly with the scheduled activities. Therefore, the lower bound value calculated for a specific node cannot be better than the values calculated for nodes existing in the path reaching to the root node. In all paths starting from the root node to the leaves (lowest level nodes) the lower bound value increases or remains the same but cannot decrease. The leaf that is guaranteed to have the lowest bound value is the optimal solution.

Branching Strategy: Starting from the root node, new branches and new nodes are formed. As stated above, root node has only critical activities scheduled. Among the remaining activities one of them is selected according to activity selection criteria, and then, new nodes as many as the number of available float days are formed. In each of these newly formed nodes, the selected activity is scheduled to one of its possible start dates and the corresponding lower bound is calculated. Since the first non-critical activity has been taken into account, this node is called the first level or first generation node. Then, similarly, branching continues for the next levels. Branching goes on from the node that has the minimum lower bound value among all open nodes as has been offered by Zamani (2001). When branching process approaches the lowest level and the newly formed nodes are leaves, the best leaf is compared with the minimum lower bound node. If the best leaf has smaller or equal objective function value than the node with minimum lower bound, this leaf is said to be the optimal solution and no more branching is required as the remaining open nodes do not have the capability to yield a better solution.

Upper Bound Calculation Strategy and Pruning: Before starting the tree formation procedure, an upper bound value is calculated. Upper bound value is nothing but just the objective function value of the best feasible solution known at any instant of the procedure. Initially, it is calculated using some simple heuristics to be used until first leaf is generated. Upper bound is used for pruning branches during the solution procedure and it is very important to reduce the solution space to deal. As new nodes are generated and lower bound values are calculated, these lower bound values are compared with the upper bound. If any

node having a lower bound value greater or equal to the upper bound value, it is said that this node does not have the ability to yield better solutions than the best known solution. Therefore, those nodes can be fathomed and no branching goes on. The branches that would be formed through those nodes are pruned. When a newly generated leaf contains better objective function value than the current upper bound value and this leaf is not proven to be optimal yet, the upper bound value is updated to be the value found in the leaf being the current best solution.

The performance of the branch and bound algorithm highly depends on the initial upper bound value and the lower bound calculation strategy. The upper bound value defines the limiting value for the lower bound values to be kept in memory. On the other hand, lower bound value is the one that is compared with the upper bound value and used to decide for the node to be kept in memory or to be deleted. In Figure 4.1 a sample solution tree has been shown. In this solution tree, 4 non-critical activities (A, B, C, and D) are scheduled. Upper bound value is assumed to be 330 and branches have been cut for values exceeding upper bound. In Figure 4.2, the simplified schematic of the solution tree has been presented to show the effect of pruning. Lower bound values shown in the nodes are hypothetical but in actual solution trees it is not much different than that representation. Upper bound value is calculated at the beginning of the branch and bound procedure by a heuristic method and it is updated during the solution. However, each newly formed node has been calculated a lower bound. That means, lower bound computation module is run as many times as the number of generated nodes. The following section is going to discuss the storage memory and the computation time dependency of the branch and bound procedure based on the quality of upper bound value and lower bound calculation strategy mentioned in this section.

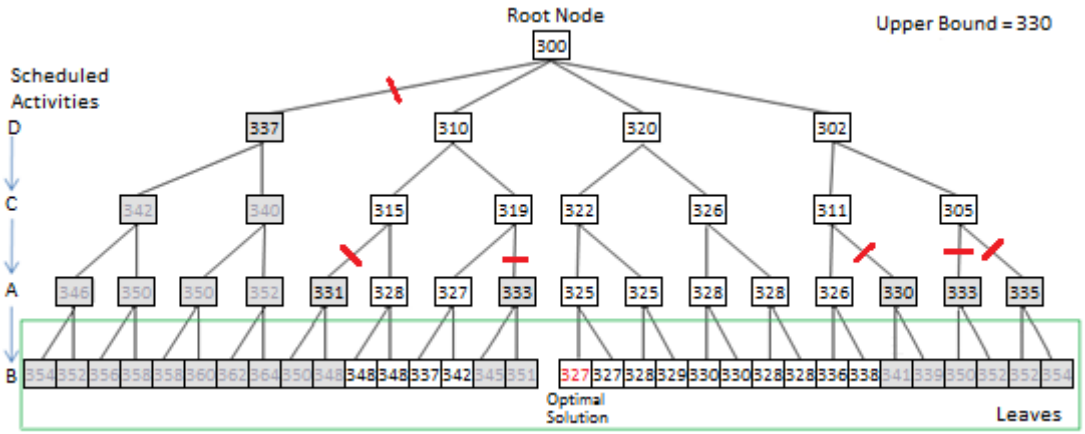


Figure 4.1 A sample solution tree of branch and bound procedure

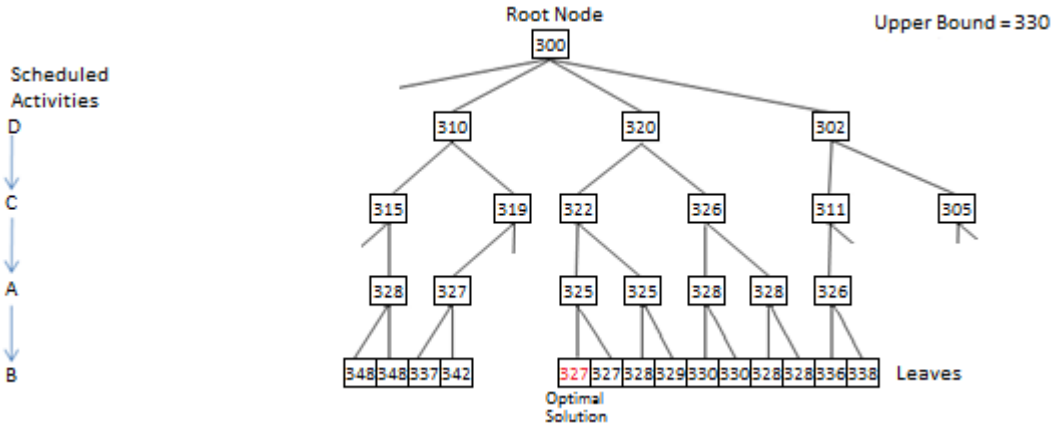


Figure 4.2 Pruned form of the sample solution tree in Figure 4.1

4.2 Storage Memory and Computation Time Dependency of the Branch and Bound Algorithm

The memory used during the solution procedure should be as low as possible not to be stack with the limit of the computer RAM capacity during operation. On the other hand, the computation time is the main parameter used for performance comparison and should be as short as possible to increase the capability of the algorithm to solve larger problems in predefined time limits.

Nodes in the solution tree are the only significant memory consuming elements. They keep the information of the start date of the scheduled activity, the lower bound value and some other information. For this thesis study, minimum data are coded to the node structure so as to ensure the minimum sized data storing element. As computer memories have their own limits, the number of nodes to be stored in the memory is limited. The node structure defined in the proposed algorithm has a size of 20 bytes roughly. That number implies that a solution tree of 50 million nodes can be handled with a free RAM memory of 1 GB. Therefore, the algorithm should be capable of reaching the optimal solution through the number of nodes filling up the available free RAM memory of the computer used for experimentation. As a result, for efficient use of storage memory, the number of nodes should be reduced as much as possible. This is possible by fathoming nodes and pruning non-promising branches. An upper bound value as good as possible and a lower bound calculation strategy as tight as possible can achieve this purpose. The quality of upper bound value and the lower bound calculation strategy determines the number of stored nodes and thereby the storage memory to be occupied.

There are several functions operating inside the branch and bound algorithm. Most of the functions are called at specific instances during the solution process while some of them are run frequently and compose the highest portion of the operation time. It has been observed that, the module that calculates the lower bound value has the most effect for computation

time. There is a trade-off with this function. For each newly generated node, this function has to be called, which means that the number of generated nodes determines the calling frequency of the function. As stated above, a tight lower bound calculation strategy leads more fathomed nodes and reduces the number of nodes to be formed in the solution tree. This procedure reduces the frequency of the function. Nevertheless, the function itself should be simple. Otherwise, as more complexity is added to the function to make it find tighter lower bound values, its individual computation time will significantly increase. In this case, the increase in the function duration will overpass the recovery gained from the reduction in the function calling frequency. To sum up, for computation time efficiency of the branch and bound algorithm, lower bound calculation strategy should be constructed in a sense that tight values are found by non-complex simple calculations.

For a complete solution procedure, the distribution of the occurrence of lower bound values calculated for nodes gives a lot of information about how to approach to the solution procedure. In Figure 4.3, a sample distribution is shown. Calculated lower bound values are in a range from the first calculated value (the root node value) to a value with the worst schedule. Optimal value is the lowest objective function value of a leaf i.e. of a complete schedule. Upper bound value can be in the range above the optimal value.

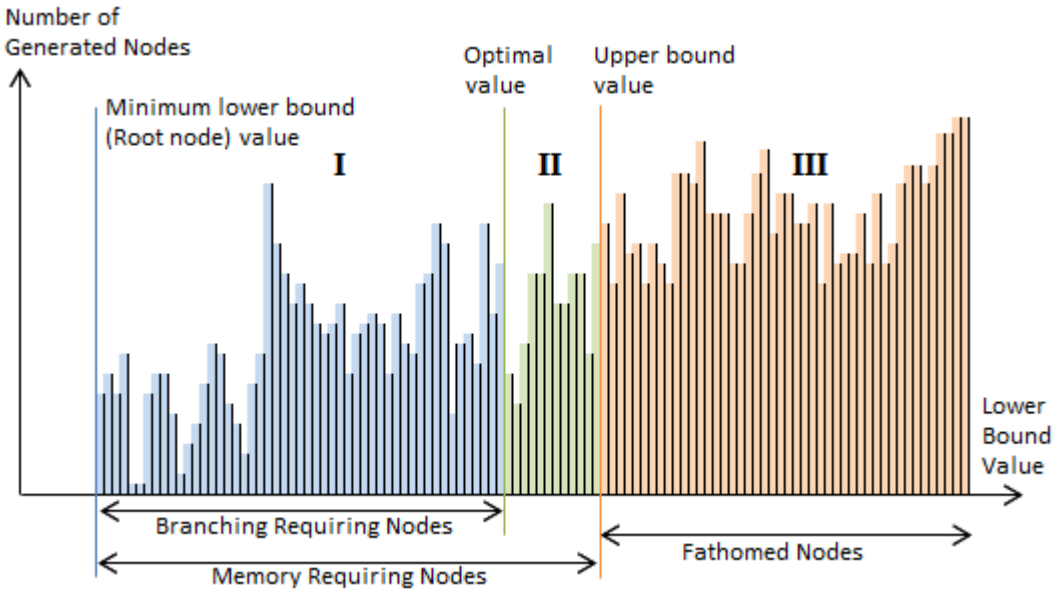


Figure 4.3 A sample distribution of occurrence of calculated lower bound values and regions that require different treatments during solution procedure

From the distribution of occurrence of lower bound values a number of propositions can be deduced:

- All nodes that have smaller lower bound value than the optimal value (Region I) must be branched i.e. must be used to generate new nodes since they still promise to lead to an optimal solution.

- All nodes that have smaller lower bound value than the upper bound value (Region I and II) must be stored in memory for branching process since optimal value is not known during the solution procedure.

- All nodes that fall above the upper bound value (Region III) can be deleted since they have lost the ability to lead to an optimal solution.

- Only the leaf with the best objective function value is kept in memory as being the current best solution and the rest is deleted. Upper bound value is updated to the leaf value if the leaf has a better objective function value.

- To avoid branching the nodes of region between the optimal value and upper bound value (Region II), the node with the minimum lower bound value among all open nodes should be selected for branching. This way, before starting to branch from this region optimal solution will already be found.

- Whenever the next node for branching has a lower bound value equal or greater than the current best solution i.e. the upper bound, the solution process is terminated and the current best solution is reported as the optimal solution.

This analysis provides us some hints on which points should be focused on to improve the storage memory and computation time performance of the branch and bound algorithm. The number of nodes falling to the branching region (Region I) and memory requiring regions (Region I and II) should be reduced as much as possible. Employed function modules should be simple as much as possible. The deductions inferred from the analysis can be listed as in the following:

- A lower bound calculation strategy with non-complex computations is required to obtain tighter values (values greater the optimal value are of interest). As more number of nodes will fall above the optimal value region, number of branching requiring nodes will go down. Only the most promising nodes will stay in the branching region and will require new node generation process. Constructing a tighter lower bound calculation strategy implies that the level of optimality promising is refined. Simplicity and tightness of the lower bound calculation strategy will determine the timing performance of the algorithm.

- A near optimal initial upper bound value would be enough to limit the memory requiring nodes. The heuristic method employed for the initial upper bound calculation should be designed such that it should be simple not to consume much time and successful to find near optimal results. The quality of the initial upper

bound value and the tightness of the lower bound value calculation strategy will determine the memory performance of the algorithm.

The techniques applied to improve the branch and bound algorithm and increase its capability to solve larger problems are mainly concentrated on the points that are inferred from the frame study stated above. Apart from the abovementioned points, algorithmic and code based improvements are also applied. All the detailed work of the developed branch and bound algorithm will be given in Chapter 5.

CHAPTER 5

AN EFFICIENT BRANCH AND BOUND ALGORITHM FOR THE RESOURCE LEVELING PROBLEM

Several improvement techniques are implemented to achieve an efficient branch and bound algorithm for the RLP. With the applied techniques, improvements for storage memory and computation time performance have been achieved. In this chapter, the developed algorithm will be explained in detail. The first section will include the definitions of the terminology of the algorithm. In the second section, the problem will be defined and scheduling methods will be explained. Third section will focus on the upper bound calculation technique and following this section lower bound calculation strategies are going to be explained in detail. In the next section, the branching strategy, node gender decision system, candidate mother queue arranging method are going to be presented and finally the flow chart of the algorithm will be illustrated.

5.1 Definitions of the Terminology of Branch and Bound Algorithm

This section included definitions of the terminology used to describe the branch and bound algorithm.

Solution tree:	The tree like scheme that represents all of the solution procedure is called solution tree. It contains the nodes and the branches that connect the child nodes to their mother nodes.
Node:	A partial solution packet that contains information of scheduled activity and the lower bound value of the partial solution is called node. In nodes one more non-critical activity has been scheduled at each generation process. The ID of that recent scheduled activity, the defined start time, the lower bound value, the mother node address and the next best node address are coded in a node element.
Root: (start node)	The first node of the solution tree is called root node. It has no non-critical activity scheduled. Thus, it has the minimum lower bound value.
Leaf: (end node)	The last node of the tree is called leaf. It has all non-critical activities scheduled to a definite start time. It contains a complete solution i.e. a feasible schedule.
Branching	Opening new branches from an existing node, that is, generating new nodes from a current node is called branching.

Mother node:	The node that has been used for branching and generating new nodes is called mother node. Root node does not have a mother node.
Child node:	The node generated by branching a mother node is called child node. By scheduling one of the unscheduled non-critical activities during branching child nodes are formed. Root node is not a child node.
Sister nodes:	Nodes of the same mother node are called sister nodes.
Candidate mother node	All nodes that are capable of generating new nodes are called candidate mother nodes and kept to produce child nodes at a later time. Once it has been used to generate child nodes, it is not a candidate anymore.
Node gender	During branching, one of the unscheduled non-critical activities is scheduled in addition to the non-critical activities scheduled up to the mother node. The ID of the additional non-critical activity that is going to be scheduled at that node is called the gender of the node. Child nodes of the same mother node have the same node gender. It is determined for each branching process by the node gender decision system.
Node gender decision system	This system decides which non-critical activity to use in branching process of candidate mother node. The selected non-critical activity highly affects the increase in the lower bound value of child nodes compared to the lower bound of mother node.
Node level (Generation or branching level)	Starting from the root node, non-critical activities are scheduled step by step as approaching to the end of the tree. The number of non-critical activities that have been recently scheduled represents the node level or the generation level. A leaf has maximum level as it has all non-critical activities scheduled to a definite start time.
Critical resources	Resources of critical activities are called critical resources.
Scheduled resources	Resources of non-critical activities that have defined start times are called scheduled resources.
Unavoidable resource periods	Periods of which resources of an activity will occur inevitably are called unavoidable resource periods. It is the period between late start time and early finish time unless late start time is later than early finish time for an activity. Otherwise, no unavoidable resource periods exists.
Unallocated resources	Resources of non-critical activities that do not have defined start times yet are called unallocated resources. They are subjected to be used for lower bound calculation. According to the lower bound strategy, they can be split to single resources or bounded to stay together.

Some definitions are represented in the Figure 5.1, where the solution tree of the problem of Easa (1989) has been presented. The upmost node (A) of the tree is seen as the root node, while at the end of the tree, leaf nodes (E, F, G, H and from K to Q) can be seen. The leaf G is the optimal solution. The numbers on the right down corners outside the nodes shows the sequence of node formation. Nodes with dotted frame (from I to Q) are the nodes subjected to pruning. In fact they are not produced in the actual procedure, but they are included in Figure 5.1 to show the complete solution tree. The solution tree has three generation levels in total. In the first level activity 4; in the second level activity 5; and finally in the third level activity 3 have been scheduled. In Figure 5.2, the change in the candidate mother node queue can be traced. As can be seen in the figure, the dotted frame means that this node is

used for branching, it is not a candidate anymore, and hence, it is omitted from the queue. At the last level the child nodes of D are all leaf nodes, and since the optimal solution is found at that moment B is not used for furthermore branching. Each newly generated node is placed in the queue using its lower bound value. Nodes with smaller and equal lower bound values take the front place. Lastly generated node has higher priority in the queue.

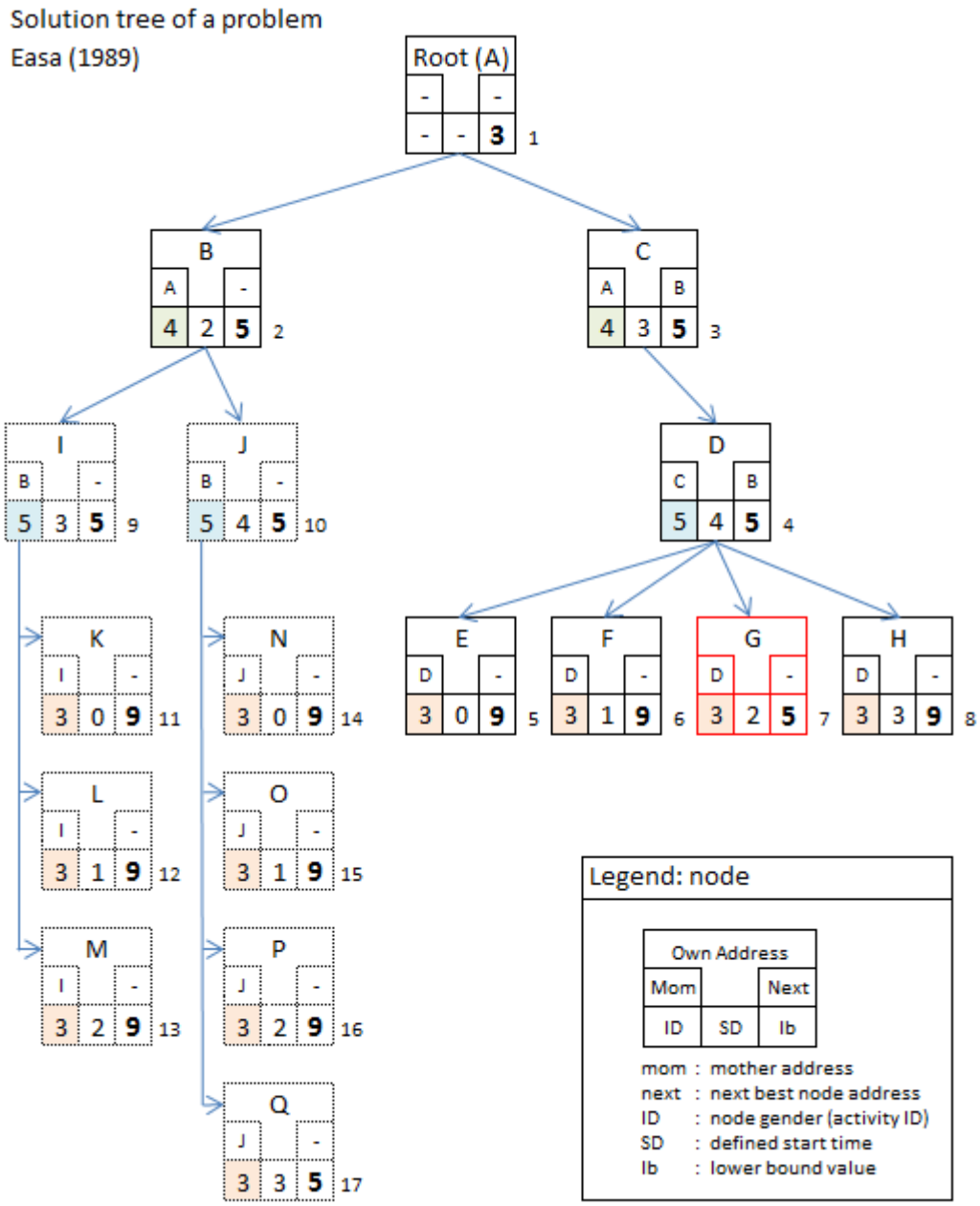


Figure 5.1 Solution tree of problem of Easa (1989)

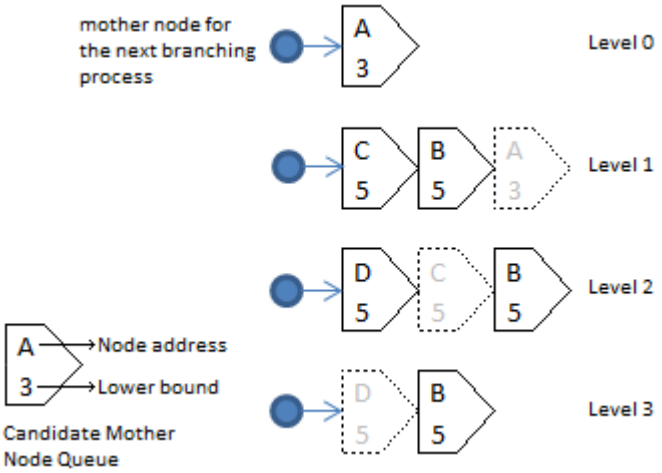


Figure 5.2 Change in the candidate mother node queue for the problem of Easa (1989) for solution tree shown in Figure 5.1

Resource utilization profile	The discrete graph of daily resource utilizations over the project duration that represents the fluctuations and leveled periods is called the resource profile. Each resource type has its own resource utilization profile.
Critical profile	The resource utilization profile obtained as a result of resource allocations of critical activities only.
Scheduled profile	The resource utilization profile obtained as a result of resource allocations of critical and scheduled activities and as well as unavoidable resource periods.
Nodal profile	The resource utilization profile obtained as a result of resource allocations of unallocated resources by distributing them over the scheduled profile.
Daily maximum allowable profile	The resource utilization profile representing the daily maximum allowable resource demands. It is obtained by resource allocations of critical and scheduled activities, and adding the daily resource demands of unscheduled activities from their current early start times to their current late finish times.
Lower bound and Lower Bound Calculation (LBC) strategy	Each node has a number of non-critical activities scheduled to definite start times and the remaining non-critical activities are not scheduled yet. The scheduled activities have their resources allocated while others have not been allocated yet. Unallocated activities are allocated with or without any constraints to obtain the best possible resource utilization profile. The objective function value of that profile is the lower bound value corresponding to the partial schedule kept in that node. The constraints employed during the allocation of unallocated resources define the lower bound strategy.

Lower bound calculation module	This module has mainly the duty of distributing the unallocated resources over the scheduled profile to obtain possible best profile while employing some strategies with constraints to tighten the calculated values for nodes as much as possible. The strategies should guarantee to yield non decreasing values as node level increases in a path from the root to the leaf.
Upper bound	The objective function value of the best possible schedule known before and during the formation of the solution tree defines the upper bound value. Before the solution procedure starts, it is found by a simple heuristic procedure. During the solution tree formation, when a leaf is produced and if it contains a better complete schedule than the best schedule at hand, the best possible schedule is upgraded to the schedule found in leaf and the upper bound value is updated to the new value.
Promising node	The nodes that are capable of leading to an optimal solution are promising nodes. A node is called promising if it has a lower bound value smaller than or equal to the upper bound value during the tree formation process.
Non-promising node	The nodes that are not capable of leading to an optimal solution are called non-promising nodes. They can be identified as the ones having greater lower bound value than the upper bound value.
Candidate mother node queue	All promising nodes are arranged to form a queue. The arrangement is performed according to the lower bound values of the nodes. The node with the minimum lower bound is the head of the queue and the immediate next node to be used for branching. The end of the queue is the node with maximum lower bound value but not greater than the upper bound since greater ones are non-promising and deleted.
Candidate mother node queue arranger module	Newly formed child nodes are also candidate mother nodes unless they are leaves. They are sent to the queue arranger module. This module places the new candidate mother nodes to the queue according to their queue indexes.
Pruning	After a branching process, the newly formed nodes checked and promising nodes are sent to the candidate mother node queue, but the non-promising nodes are deleted. The branch of non-promising nodes is pruned in the tree to simplify the solution procedure.

A partial solution packet taken from the solution procedure of problem *rlp_10_1_1* from test set T_2 of Rieck et al. (2012) has been presented in Figure 5.3. Critical, scheduled, unavoidable and unscheduled resources have been shown. Note that activity_10's early start time has been delayed due to the scheduled start time of activity_2 and the temporal relation in between. Therefore, an unavoidable resource period has been occurred (duration of activity_10 is 7 days). Unallocated resources are distributed one by one within the daily maximum allowable resource usage profile to obtain a lower bound. The cost of the resources is given as 4, and optimal solution results of that instance in case of future study needs are as follows: SSQR: 2472, OVERLOAD: 28, RID-MRD: 96.

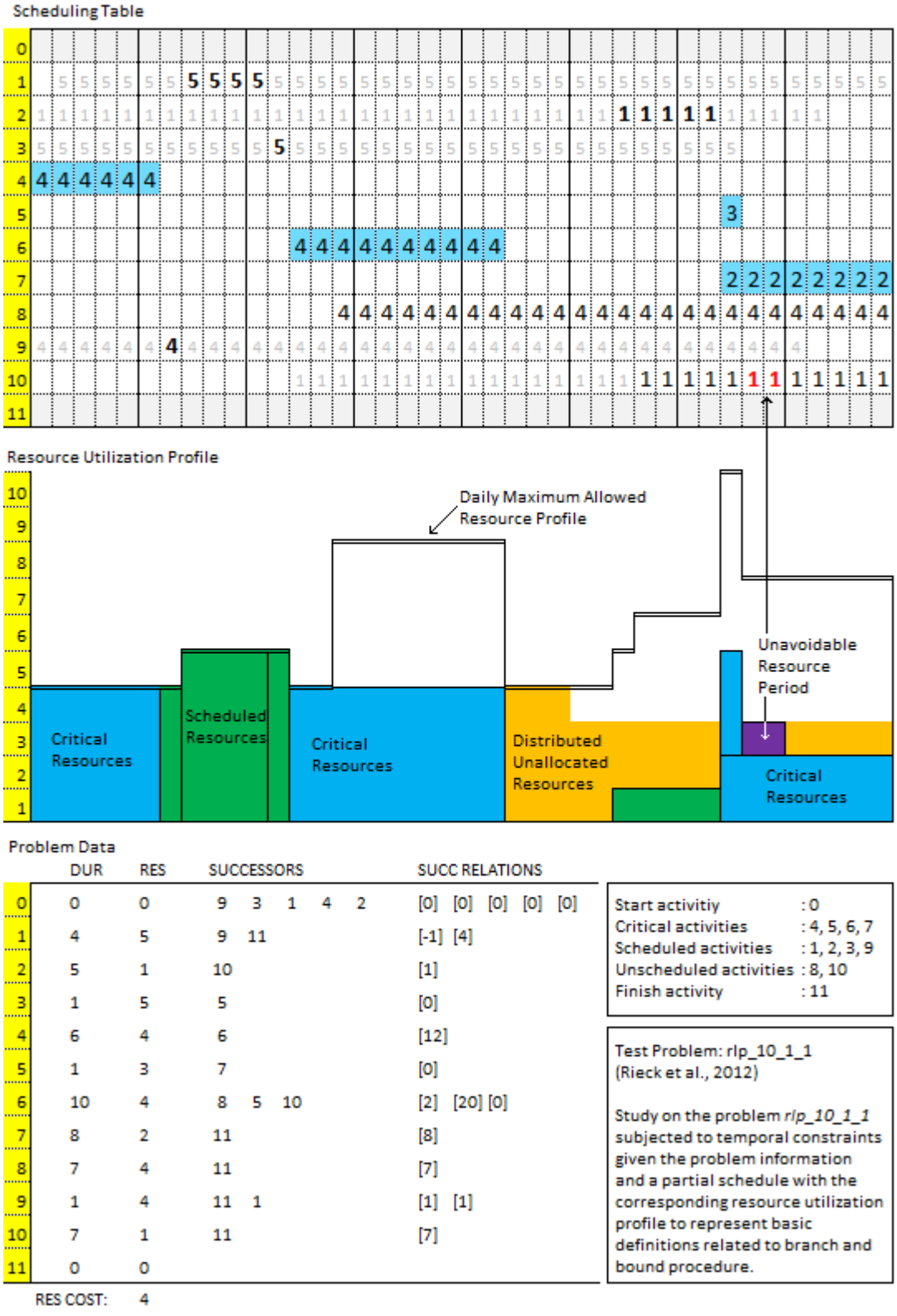


Figure 5.3 Representation of basic definitions using problem of *rlp_10_1_1* from test set T_2 of Rieck et al. (2012)

Queue index	Arrangement of candidate mother node queue is carried out by indexes. The size of the indexing is the difference between the initial upper bound value and the root node lower bound value. Each node has a queue index calculated using its lower bound value. Candidate mother nodes are placed to the queue according to their indexes.
Schedule map	A list that shows which activities have been scheduled and to what start times and which activities have not been scheduled yet. Before a branching process takes place, it is determined by tracing and picking information from the mother nodes of the current candidate mother node up to the root node.
Rescheduling	When scheduled non-critical activities have been defined start times that may affect start and finish times of the unscheduled activities. Recalculating early and late start times of unscheduled non-critical activities regarding the current schedule map is required to eliminate possible infeasible schedule variations.
Rescheduler module	This module utilizes the schedule map and performs scheduling over the activities by treating the start time defined activities as fixed. The non-critical activities that have not defined start times yet are calculated for their new early and late start times and then, total floats are updated.

5.2 Problem Instances and Scheduling Methods

In project scheduling, there are certain types of relations among tasks. The relationships represent the work logic. The simplest way of defining a work logic is having precedence relationships between the activities. The general way of defining all relationships between the activities is provided by temporal constraints. In this thesis study problem instances subjected to both precedence relations and general temporal constraints have been used. Problem instances with precedence relations are scheduled using critical path method (CPM). While on the other hand, instances subjected to general temporal constraints are scheduled using temporal scheduling (TPS). In this section, both scheduling methods are going to be explained.

5.2.1 Problems subjected to precedence relations and CPM scheduling

The simplest way of definition of task relations is the precedence relations. In this relation, tasks are scheduled to start after finish of their predecessors. Once the predecessor of the task is finished, it can be started any time without causing any delay at the project completion time. The procedure of CPM scheduling contains two processes, one is forward pass to find early times and the other is backward pass to find late times. In forward pass, start activity (which is generally a milestone) is started at time zero. Then, the activity which has all its predecessors' early start times calculated is started to the latest early finish time of its predecessors. This way all activities' early start times are calculated. The finish activity is started at the latest finish time of its predecessors and this way forward pass is completed. In backward pass, finish activity is assigned a late finish time at its early start time as it is a dummy activity representing the finish of the project. Similar to the forward pass, the activity having all its successors' late finish times are calculated is finished to the earliest

late finish time of its successors. This way all activities' late start times are calculated. After early and late start times are calculated for each activity, available float for activities can be computed easily by subtracting the early start time from late start time. The activities having zero float are critical activities and compose the critical path of the project.

The considerations in precedence relations and CPM scheduling are relatively simple. In Figure 5.4 and 5.5, the activity network diagram (AoN) and early start schedule and an optimal schedule with respect to SSQR metric with corresponding resource utilizations are presented. Relationships are given as FS (finish to start) with zero lags. Note that, the early start schedule corresponds to leaf (end node) K, the optimal schedule corresponds to leaf G in the solution tree given in Figure 5.1.

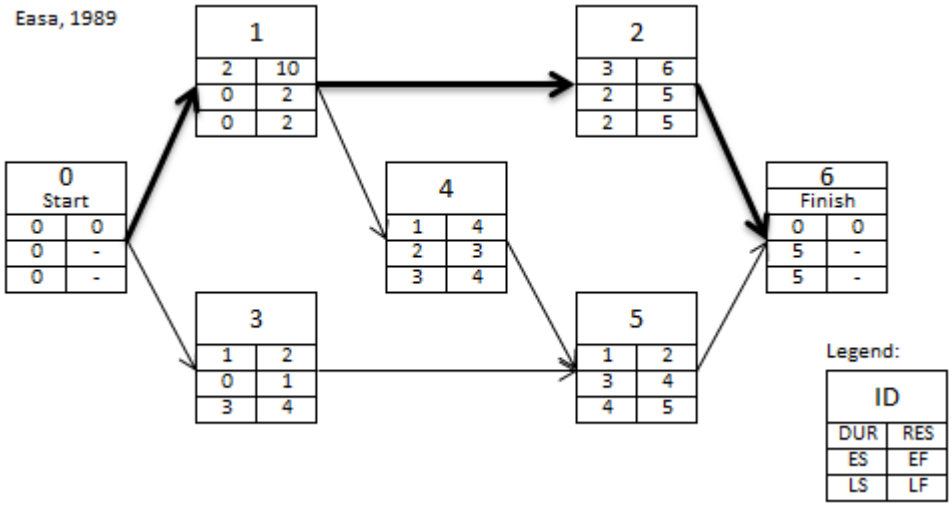


Figure 5.4 Activity network diagram of problem of Easa (1989) with precedence relations

5.2.2 Problems with general temporal constraints and TPS scheduling

A more general way of representing activity relations is achieved under general temporal constraints (Neumann et al., 2003). In this relationship type, there are minimum and maximum time lags between the start times of the activities. Similar to the precedence relations case, there is a critical path, however, the critical path is neither continuous nor complete. That is, there may be periods of which no critical work is performed (see Figure 5.3 – Scheduling Table) or first critical activity may start at the middles of the project. All relationships depend on the tasks' dependencies under various conditions. Under general temporal constraints, relationships are defined based on start times as time lags. Activities may have minimum time lags, maximum time lags or both at the same time dually. Instances subjected to general temporal constraints are scheduled using temporal scheduling.

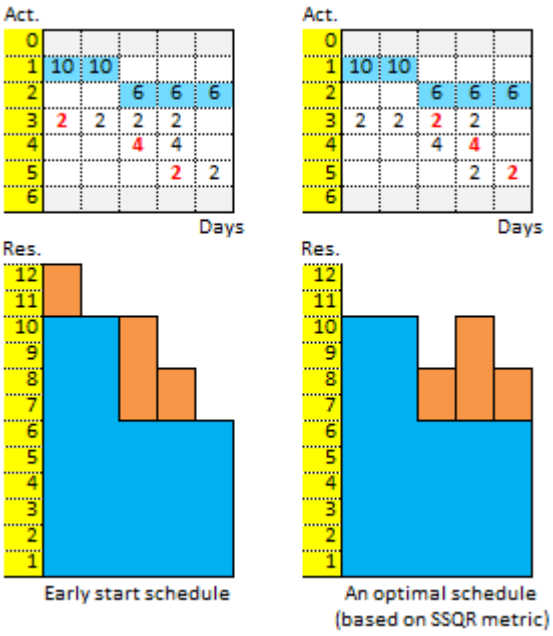


Figure 5.5 Early start schedule and an optimal schedule for problem of Easa (1989)

Temporal scheduling is very similar to CPM scheduling and has forward pass for early start times determination and backward pass for late start times determination. Before starting to forward pass, all activities' early start times are initialized at time zero. Then, each activity's early start time is checked if it satisfies the minimum and maximum time lag relationships with its predecessors' early start times. If it does not satisfy, the early start time of the activity is shifted forward to the earliest start time that satisfies all temporal relations with its predecessors. This way, early start times of all activities are calculated and the early start time of the finish activity is assigned as the project due time. Forward pass calculations are iterated as many times as the number of activities so as to ensure the effect of a change in an early start time of a predecessor is taken into account. Before starting to backward pass, all activities' late start times are initialized at the project due time. Then, similar to forward pass, each activity's late start time is checked if it satisfies the minimum and maximum time lag relationships with its successors' late start times. If it does not satisfy, the late start time of the activity is shifted backward to the latest start time that satisfies all temporal relations with its successors. This way, late start times of all activities are calculated. Backward pass calculations are iterated as many times as the number of activities so as to ensure the effect of a change in a late start time of a successor is taken into account. After early and late start times are calculated, available floats can be computed easily. Early and late finish times of activities can be found by adding the duration of each activity to their start times. Similar to the CPM case, activities with zero total floats are called as critical activities.

The network diagrams of instances subjected to general temporal constraints may have double arrows defining the relationship between two activities. When double arrow case occurs, it means that those dual activities have both minimum and maximum time lag

relationships. Figure 5.6 represents the activity network diagram (AoN) of problem *rlp_10_1_1* (Rieck et al., 2012). Time lags are noted on the corresponding arrows. Since scheduling calculations are carried out on start dates and finish dates can easily be calculated later using start dates and durations, only start dates are noted in the diagram. A sample partial schedule and corresponding resource utilization profile of *rlp_10_1_1* used for resource leveling also has been presented in Figure 5.3.

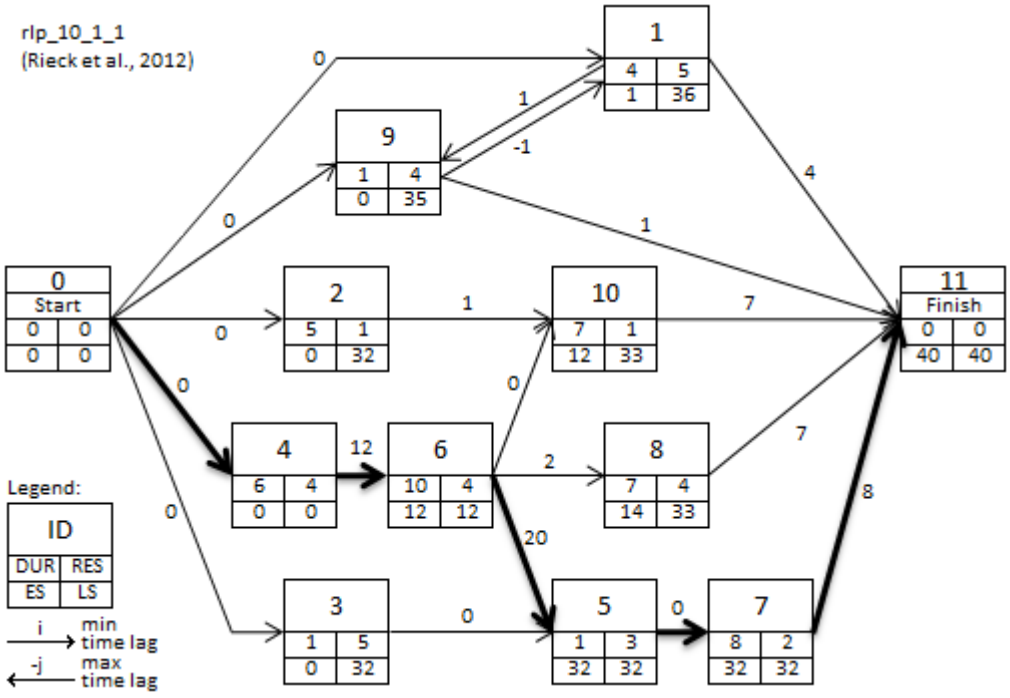


Figure 5.6 Activity network diagram of problem *rlp_10_1_1* (Rieck et al., 2012) with general temporal constraints

5.3 Adapted Branch and Bound Heuristic for Upper Bound Calculation

An upper bound value defines the limit of the objective function value for the nodes to be kept in memory and it is an effecting parameter in determination of storage memory performance. It has been stated before that upper bound is the objective function value of the best known solution. Before starting the branch and bound procedure for the exact solution, upper bound is found by solving the problem with a heuristical method. Then, throughout the procedure, it is updated whenever a better solution is found. An adapted branch and bound heuristic has been developed to determine a good upper bound value. The heuristic to calculate upper bound should be simple as not to require much computation time, and

should be successful as to yield near optimal results enough to limit excessive memory usage. For this purpose, already studying branch and bound algorithm for exact solutions of RLP, with some modifications an adapted procedure has been proposed for heuristic solution.

The adapted branch and bound heuristic differs from the ordinary branch and bound procedure in terms of node gender decision system, number of iterative procedures and promising node acceptance criteria. Node genders, i.e. activity ID's to be used for branching, are predefined according to some strategies. There are two main iterative procedures; void filling and grading cycle and rehabilitation cycle. Each cycle has a solution tree grown up. Finally, a promising node acceptance criterion has been defined in duty of upper bound. The details of the adapted branch and bound algorithm are explained as in the following.

5.3.1 Node gender decision system – Activity priority determination

In node gender decision system, activities to be used for branching are identified at the beginning. For each node level, the activity ID to be used for branching is defined. The activity to be used for branching during tree formation has already been stated to affect the increase or change in the lower bound. In fact, most effecting activities are preferred to be branched prior to others. Activity priority determination in simple heuristics is done according to the activity ID, total float, resource demand, duration etc. During early experimentations, it has been realized that the criteria defined for the activity priority is highly problem dependent. That is, while for some problems, ascending total float is very effective, it may deteriorate the performance for some other problems where descending resource demand is the top effecting parameter. In order to find the best effecting priority for branching procedure, some high effecting parameters have been defined and several activity priorities have been applied. The problem is solved regarding all activity priorities and the best result is picked and reported. The parameters used in the determination process are as follows:

- (resource demand)*(duration)
- (resource demand)*(unallocated duration)
- (resource demand)
- (total float)
- (non-critical activity crowdness)

where (unallocated duration) = (duration – unavoidable period) and non-critical activity crowdness for an activity is the number of non-critical activities that may be performed at the same time with that activity. Considering ascending and descending orders of these parameters, ten different sequencing criteria are obtained. A further two sequencing stages are applied in order to arrange the activities that have equal value according to the selected criteria at the first stage. In the second and third stages the same ten criteria has been applied

for sequencing. Figure 5.7 illustrates the activity sequencing strategy employed in the developed algorithm and the best priority selection.

At the first stage, activities are sequenced according to the parameters and the heuristic procedure is run for each procedure. The priority which has yielded the best result is selected and for the second stage it is used for the first level sequencing. Activities that have equal values in the first level (for example, activities having the same [res]*[unalloc dur] values) are sequenced in the second level. Similarly, activities with same values (same [res]*[unalloc dur] and [total float] values) are sequenced in the third level. Experimentations have shown that sequencing up to third level is reasonably enough. This way, the problem instance is solved 30 times heuristically and the best result is picked up through and reported. Moreover, for the first and second stage, it can be seen from the Figure 5.7 that, activities are not scheduled utilizing all levels. For the 1st and 2nd levels, intuitively chosen descending resource demand and descending total float criteria respectively are employed to enhance the results. Therefore, the first stage is carried out by [x, 2, 3], second stage by [(1st best), x, 3], third stage by [(1st best), (2nd best), x] so as at the end the [(1st best), (2nd best), (3rd best)] sequencing criteria is obtained. The activity priority determination process has been presented in Figure 5.8. For further research, new sequencing criteria can be defined and different sequencing strategies can be applied. The aim in applying multi-level different sequencing strategies is to cover up various problem dependencies for sequencing. Although the problem is solved 30 times heuristically, the computation time is negligible compared to the exact branch and bound procedure.

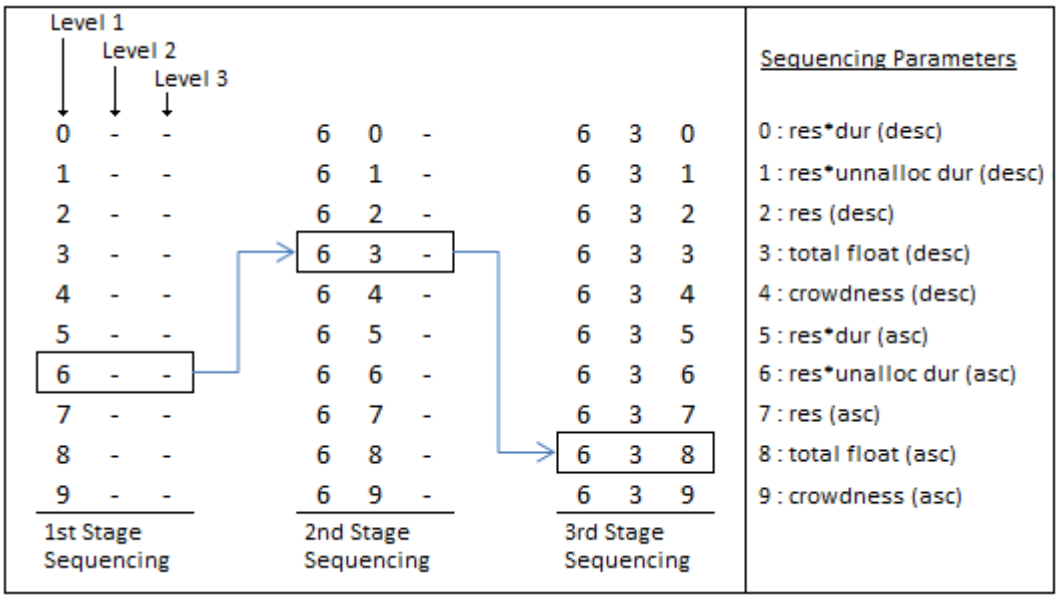


Figure 5.7 Activity sequencing and best priority selection

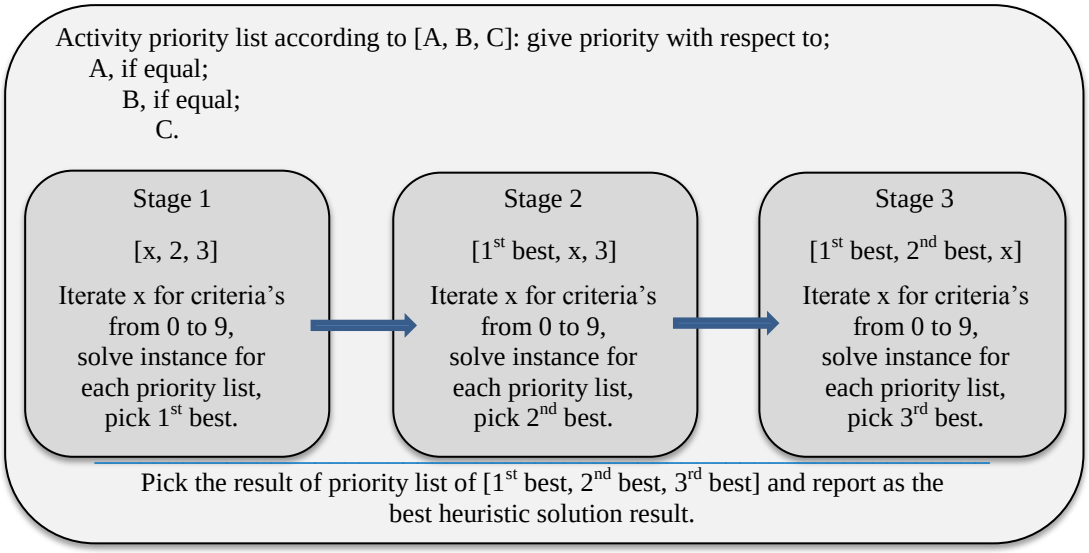


Figure 5.8 Flow of activity priority determination process for heuristic module

5.3.2 Iterative procedures of the branch and bound heuristic

As stated above, heuristic procedure has two different iterative cycles. The first cycle is void filling and grading cycle and the second is the rehabilitation cycle. In the first cycle, a primitive solution is generated. Then, rehabilitative cycles are applied on the generated solution to improve the result.

In the void filling and grading cycle, the root node is constructed by scheduling only critical activities. As generation level increases, the next activity in the priority list is scheduled. At a generation level, activities up to that level in the priority list are scheduled and the resources of remaining activities are unallocated. Unallocated resources are distributed according to their possible effects on the resource profile, that is, they are uniformly distributed to the periods that they can occur. The total number of unallocated resources is divided to the length of the period of ES-LF, and the resulted number is used to shift the daily demands of that period. After the distribution, we may come up with fractional daily demands. Figure 5.9 shows the resource utilization profile of a node during tree formation of the void filling and grading cycle of the problem *rlp_10_1_1* of Rieck et al. (2012). Note that, activity_8 and activity_10 are unscheduled activities. Their total resource demand is distributed uniformly through the range of ES-LF. The resulting resource profile has fractional daily demands. This way, the effect of unallocated activities is aimed to be taken into account. The resource utilization profile resembles to a non-uniform rough way, the resource blocks to be allocated by scheduling resembles to bricks. Allocating resources resembles like filling the voids in the resource profile. That is why it is called void filling. The unallocated resources are distributed uniformly to their occurrence range so their effect is accounted. The overall process resembles grading a surface of a road. Hence, the procedure is called void filling and grading. The resulted resource utilization profile is used

to calculate the bounding value of the node. At the last generation level, complete schedules are obtained and the best schedule is selected to be imposed to rehabilitation cycle.

In the rehabilitation process, the root node and all nodes have complete schedules. The root node schedule is the best solution found in the void filling and grading cycle. New nodes are generated by unscheduling the activity that corresponds to the branching level from the priority list and scheduling it again within its updated available float days. In each branching level, corresponding activities are rescheduled to their available float days to pick better solutions. All nodes of the rehabilitation cycle contain a feasible schedule. Rehabilitation cycle is iterated a number of times until no further improvement is obtained on the current best solution. The final best solution is the best solution obtained regarding the predefined activity priority.

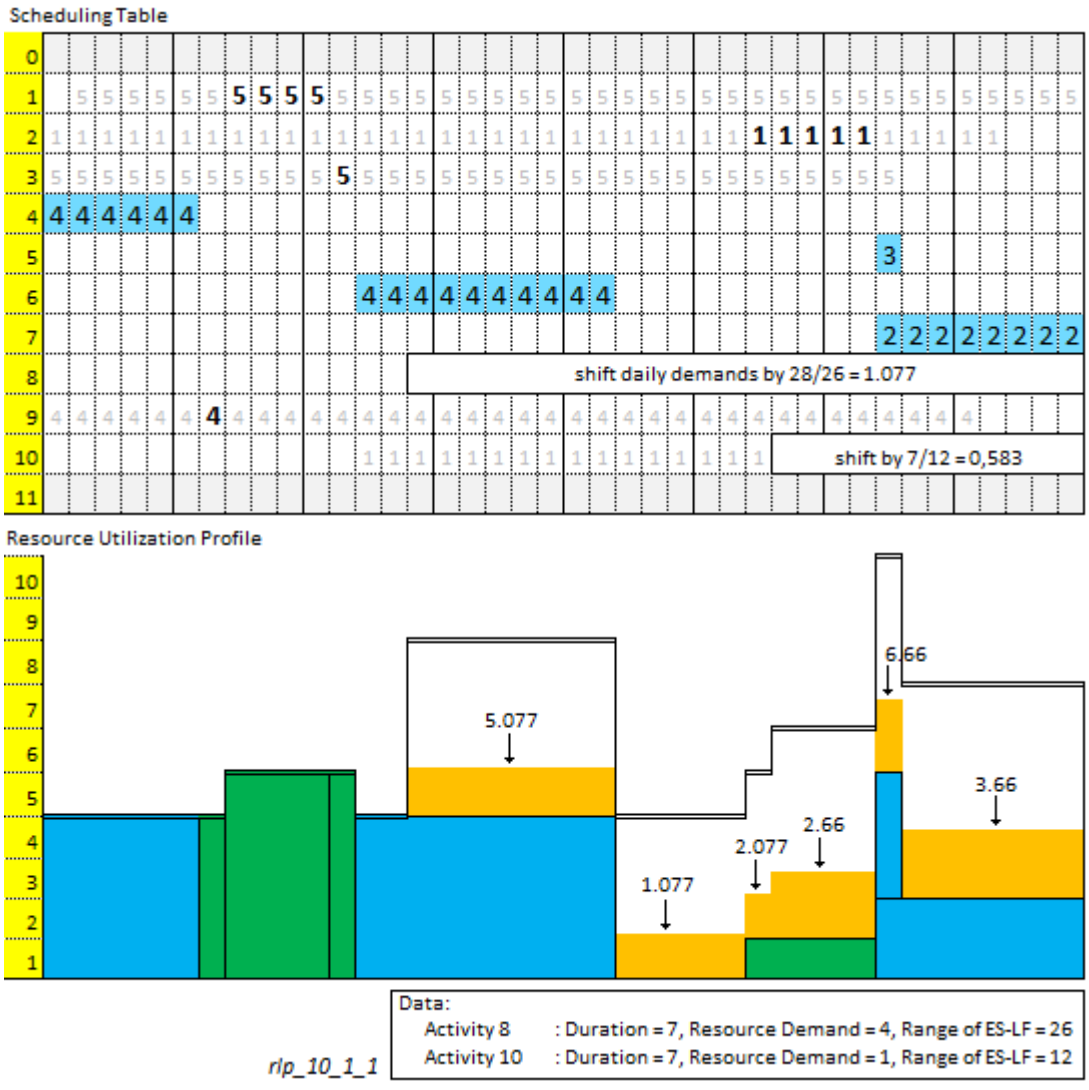


Figure 5.9 Bound calculation with void filling and grading

5.3.3 Promising nodes acceptance criteria

Similar to the ordinary branch and bound procedure, nodes are generated and a bound value is calculated in the void filling and grading cycle while objective function value is calculated for complete schedule nodes in the rehabilitation cycle. Different than the exact solving procedure, branching does not goes on from the node having the best (smallest) value, but instead it goes level by level, i.e. after one generation level is finished completely, the next generation level is started. In actual cases, the number of nodes increases exponentially. Here indeed, the number of generated nodes in a level are sequenced from best (smallest value) to the worst (greatest value) and only a number of the best nodes are accepted eligible for branching in the next generation level. All remaining nodes are omitted. Therefore, the growth of the tree is kept limited. For experimentation, this limiting number is started from 10 for the first generation level and decreased linearly to 1 up the last generation level. With this limitation, the sensitivity of keeping a best solution is aimed to be high, and at the same time, the number of branched nodes is aimed to be kept limited due to time and capacity limitations. Table 5.1 illustrates the limitation in the growth of the solution tree compared to the ordinary case of the problem given by El-Rayes and Jun (2009). The problem is of 20 activities and has 13 non-critical activities. Branching on all nodes leads to finding the optimal solution but it requires millions of nodes to process. By the node acceptance limitation, this number is reduced to just hundreds. That is actually the main reason why this heuristic does not take much computation time.

Table 5.1 Tree size growth of ordinary and limited node acceptance cases for the problem of El-Rayes and Jun (2009)

Node level	ID	(TF+1) of each activity	Length of the candidate mother node queue (ordinary case)	Limiting number of eligible nodes for the next level branching	Length of the candidate mother node queue (limited acceptance case)
$[1] = i$	[2]	[3]	$\prod_{j}^{i}[3]_j = [4]$	$10\left(1-\frac{i-1}{n}\right) = [5]$	$[3]_i * [5]_{i-1} = [6]$
1	4	5	5	10	5
2	5	2	10	9	10
3	15	2	20	8	18
4	12	2	40	7	16
5	20	6	240	6	42
6	13	5	1,200	6	30
7	18	6	7,200	5	36
8	2	5	36,000	4	25
9	17	2	72,000	3	8
10	7	5	360,000	3	15
11	14	5	1,800,000	2	15
12	9	8	14,400,000	1	16
13 = n	10	5	72,000,000	1	5
Total:			88,676,715		241

Note: Available float days of activities are assumed not to show variations due to scheduling of other activities during formation of the solution. Values given for the lengths of candidate mother node queues do not represent exactly actual values as in actual solution tree values are slightly different.

The proposed heuristic algorithm uses adapted branch and bound procedure to generate a good solution in shortest possible time. Meanwhile, applying several different activity priority lists for branching, including unallocated resources' effect for bound calculation in void filling and grading cycle and iterating rehabilitative cycles on the primitive solution to improve the result, a good upper bound value is obtained for the exact solution procedure.

5.4 Lower Bound Calculation Strategy

For the branch and bound algorithm, one of the most significant issues is how to calculate the lower bound value. The lower bound value is so important that it is the most effecting parameter for time and capacity performance of the algorithm. As lower bound calculation includes more considerations of problem variables and constraints, it yields more desirable (tighter) results. Nevertheless, more complexity introduced to the calculation strategy means more computation effort. Since lower bound calculation is performed for every node generated, any increase in computation time of the lower bound calculation module is reflected as it is to the overall computation time. In this study, a new lower bound calculation method has been developed and proposed to be used together with the existing lower bound calculation method. In the following sections, both calculation strategies have been explained, and strength of both methods have been compared. Both calculation methods have been observed to have different strengths which generated the idea to use both methods together at the same time in the procedure.

5.4.1 LBC-1 – Distribution of unallocated resources one by one method

This is the most known lower bound calculation method proposed by Neumann et al. (2000). It relies on distributing unallocated resources over the scheduled profile one by one so as to obtain the best leveled resource utilization profile. The only constraint employed in this method is that resources are distributed within the daily maximum allowable resource utilization limit.

Before starting to distribute unallocated resources, scheduled profile is obtained by allocating resources of critical and scheduled activities and unavoidable periods. Then, daily maximum allowable profile is obtained. Unallocated resources are distributed over the scheduled profile to the lowest demand periods for SSQR, ABSDEV and OVERLOAD metrics without exceeding the daily maximum allowable limits. For RID-MRD metric, resource distribution is carried out by first filling the resource idle days of the scheduled profile, and then, if unallocated resources still exists, lowest demand periods are filled. Figure 5.10 shows one by one unallocated resource distribution for lower bound calculation. Then, nodal resource utilization profile is ready to calculate its objective function.

LBC-1 calculation method has mainly three steps;

- Scheduled profile calculation
- Daily maximum allowable profile calculation
- One by one distribution of the unallocated resources.

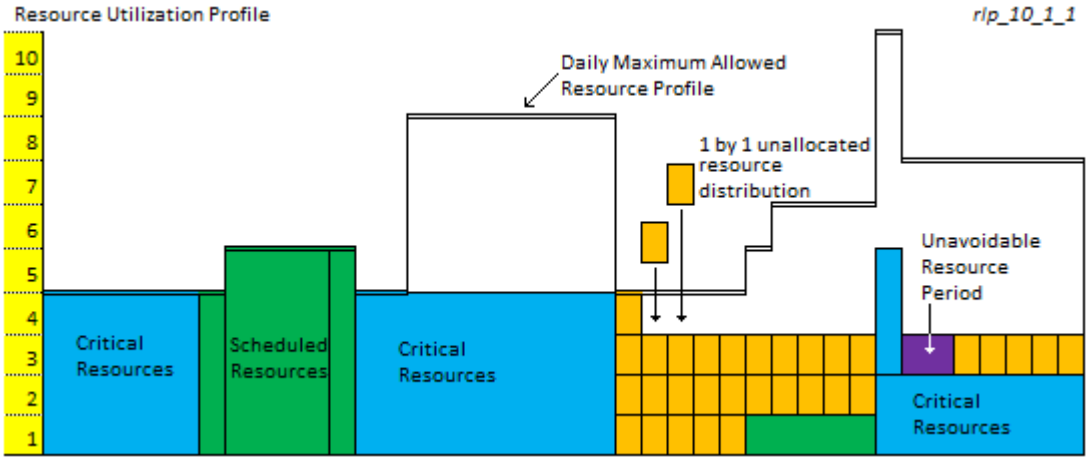


Figure 5.10 Lower bound calculation by distributing unallocated resources one by one

During branching on a mother node, child nodes are generated one by one and for each node its lower bound value is calculated. When schedule maps of sister nodes are compared, it can be observed that the only difference is the one day shift occurring at the start time of node gender activity. Since sister nodes have the same ancestor nodes, apart from the node gender activity, all other scheduled activities have the same schedule. Therefore, a further simplification can be implemented for the calculation of lower bound steps. When lower bound value has been calculated for the first child node, the scheduled profile and daily maximum allowable profile calculation data is stored. This data is used during lower bound calculation of sister nodes by applying respective modifications. Only unallocated resource distribution is repeated for each sister node. This data inheritance between sister nodes has significantly reduced the computation time shared by lower bound calculation module, and it improved the algorithm.

5.4.2 LBC-2 – Uncombined individual unscheduled activity resource effect method

This method is a new method presented in this thesis and represents a different approach for lower bound calculation. Unscheduled resources are not distributed. Instead, individually unscheduled activities are placed over the scheduled profile by trying their float day's one by one. For each trial, the minimum resource effects of individual activities are summed up and accounted for the lower bound calculation. Resource demand of an unscheduled activity is not split and it is considered within its available schedule periods. The quality drawback in this method is that unscheduled activities are not considered with their combinatorial effects during resource effect calculation.

Similar to the LBC-1 strategy, scheduled profile is calculated at first and it should be ready. Daily maximum allowable resource profile is not required in this method. Then, unscheduled activities are determined. One by one, each activity is taken to the resource effect calculation process. In this process, the unscheduled activity is started at each day of

its available float days. For each case, the resource effect according to the objective function is calculated. Minimum resource effect is picked and noted as the unscheduled activity's individual resource effect. All minimum resource effects are summed up and added to the objective function value of the scheduled profile. The final obtained value is coded as the lower bound value to the node.

In Figure 5.11, uncombined individual resource effect calculation is illustrated using the resource utilization profile of *rlp_10_1_1* of which schedule has been mention earlier in Figure 5.3. Activity_8 and Activity_10 were the unscheduled activities. Each activity is scheduled to its available float days individually. Objective function is calculated for each time, and the minimum increase compared to the scheduled profile objective function value is taken as the resource effect of that unscheduled activity. Resource effect calculation of an unscheduled activity does not affect other activity's resource effect calculations.

LBC-2 calculation method has mainly two steps;

- Scheduled profile calculation
- Uncombined individual unscheduled activity resource effect calculation.

LBC-2 is relatively much more complex compared to LBC-1. It requires much computation time to calculate resource effects for each node. Even considering sister nodes, one day shift at the start time of the node gender activity may cause all unscheduled activity resource effects to change. Although scheduled profile calculation data can be inherited between sister nodes, the resource effect calculation is still complex enough to take the most computation effort. Therefore, a different data heritance method has been developed with a cost of slight quality decrease. That is, the unscheduled activity resource effects are calculated based on the scheduled profile in which the node gender activity is also assumed unscheduled. The scheduled profile indeed belongs to the mother node. Uncombined resource effects of each individual unscheduled activity are calculated just before child node generation process. Even the node gender activity has been imposed to the uncombined resource effect calculation. This situation will be referred in section 5.5, where it is also related with the node gender decision system. Minimum resource effects of each individual unscheduled activity are summed up except the one of node gender activity. This summed value is used for the whole sister nodes lower bound value calculation. For each child node, objective function value of the scheduled profile is calculated and by the addition of the total uncombined individual unscheduled activity resource effect the lower bound value is obtained. This method can be called as mother data inheritance method. Considering that a mother node can generate tens of child nodes, uncombined resource calculation process is run once for mother node and never for child nodes. This method has saved from the computation time up to hundred times. It has been observed that the positive outcome overpassed the loss of the quality due to non-consideration of the node gender activity during resource effect calculation.

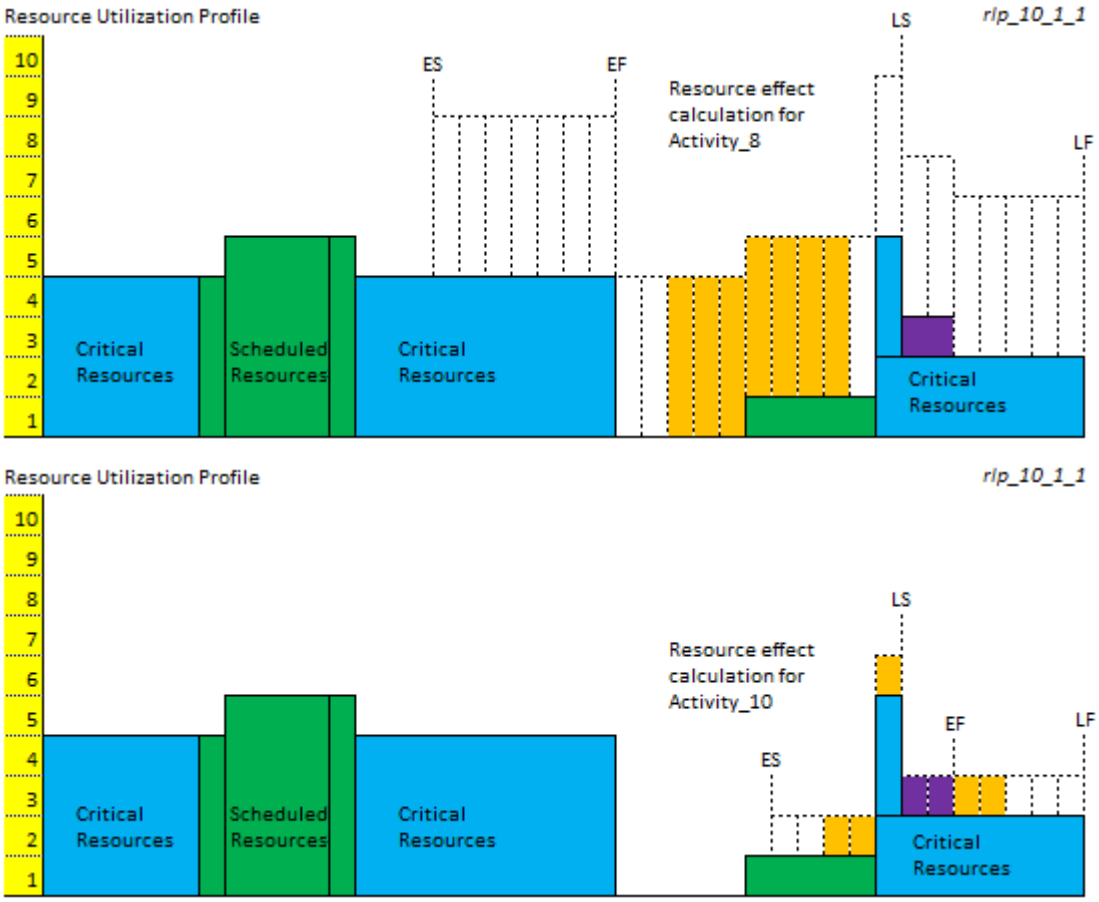


Figure 5.11 Uncombined individual unscheduled activity resource effect calculation

Comparing LBC-1 and LBC-2, it has been observed that both methods have different strengths. LBC-1 has been monitored to yield lower bound values spread to a range of moderate minimum – moderate maximum lower bound value. On the other hand, LBC-2 has been monitored to yield lower bound values spread to a range of small minimum – high maximum lower bound value. For example, using LBC-1, early generated nodes have moderate lower bound values. As more nodes are generated, the new lower bound values show gradual increase. Some of the values exceed optimal value and remain unbranched while other nodes are branched until optimal solution is reached. However, using LBC-2, early generated nodes have very small lower bound values. Yet, as more nodes are generated, the new lower bound values show fast increase. Therefore, many of the values are pushed over the optimal value and fewer nodes remain for branching. It cannot be said that LBC-1 is better that LBC-2 or vice a versa. Based on experimentation, both lower bound calculation strategies performed well for different problem sets. The situation of being strong at different parts of the solution procedure triggered the idea to use both calculation strategies together at the same time. With dual implementation, two lower bound

values are calculated for each node. The tighter (greater) value is selected and coded to the corresponding node. With this implementation, the computation time is increased (approximately doubled), but the strength of both strategies is combined. Hybrid used of LBC-1 and LBC-2 has revealed a very powerful and high performance bounding for branch and bound procedure. Larger size problems could be solved and the performance of the algorithm has shown relatively good results.

The use of LBC-2 has also enabled an effective way for implementing the node gender decision system. In the following section, how the node gender is decided using LBC-2 data will be explained.

5.5 Node Gender Decision, Branching Strategy and Candidate Mother Node Queue

Node gender, that is, the activity to be scheduled when branching from a mother node is a significant element that affects the lower bound value growth. Branching strategy also affects the path to reach the optimal solution. Finally, candidate mother nodes, which are the nodes to be branched for generating new nodes, are required to be organized well so that at a large solution tree sizes interpreting millions of nodes becomes possible. There are two three node gender decision methods employed in the developed algorithm. In the first one, a predefined priority list is utilized while at the second one, average resource effect calculation has been employed to determine node gender activity. The third one is slightly different than the first method. All methods are going to be explained in detail in the following sections. Then, branching strategy and candidate mother node queue with queue indexing are going to be explained.

5.5.1 NGD1 – Predefined priority list for node gender decision

In this node gender decision method, the activity to be scheduled for branching from a mother node is determined according to a predefined activity priority list. This list is formed before starting the tree formation procedure. The generation level of the mother node to be branched is found. Then, the activity that corresponds to this generation level from the priority list is decided as node gender. One of the sequencing criteria that have been mentioned in section 5.3.1, for node gender decision system of adapted branch and bound procedure, has been utilized to determine the activity priority list. Based on experimentations, the sequencing criteria of [2-1-8] – which employs descending resource demand for the first level, descending amount of unallocated resources for the second level and ascending total float for the third level – has performed best results in general. Activities are sequenced according to that criteria and obtained activity priority list is used for node gender decision. Some other intuitively chosen sequencing criteria have also been experimented. However, in average, other criteria have not performed as good as the above mentioned one.

In NGD1 method, the activities to be scheduled at each branching level are predefined and no calculation takes place during branching for node gender decision. However, as a drawback of that method, the selected criteria for priority determination is problem dependent. Some problem instances may require much computation time compared to the

same size instances. Besides, there is no mean to solve the problem many times with different criteria based priority lists and pick the best performance.

5.5.2 NGD2 – Maximum average resource effect calculation for node gender decision

This method has been inspired from the individual unscheduled activity resource effect calculation for LBC-2. Minimum uncombined individual unscheduled activity resource effects are summed up and added to the scheduled profile objective function value to find a lower bound value for issued node. In section 5.4.2, it has been stated that unscheduled activity resource effects are calculated using mother node's information. Just at this stage, the calculation data is utilized for average resource effect calculation also. For lower bound calculation purposes, the minimum resource effect of an unscheduled activity is picked up. On the other hand, for node gender decision purposes, all resource effects are summed up, divided to the number of possible start times (TF+1) to find average resource effects and then, the unscheduled activity with the maximum average resource effect is selected as node gender. The resource effect calculation is illustrated in Figure 5.11. Each unscheduled activity is scheduled for its possible start times starting from ES time to LS time. For each case the resource effects are summed up and then, the average resource effect is found. The unscheduled activity with the maximum average resource effect is considered the one that will cause the most increase in the lower bound values of child nodes compared to their mother node. A high increase in the lower bound value increases the possibility of generating more non-promising nodes and thereby, increases the possibility of pruning.

This method is a dynamic node gender decision method. However, since already available calculations are utilized, computation time is not increased. The activity scheduled may not be the same for the same generation levels. The idea of using maximum average resource effect plays an important role for fast lower bound tightening. This leads to pruning more nodes at early stages of the solution tree before having a large tree size. Both node gender decision methods have been tested. Experimentation results have shown that this method has proven better in average compared to the predefined priority method although for some problem instances predefined priority method showed far better performance. Test results have been presented in Chapter 7 for both methods.

5.5.3 NGD3 – Dynamic priority list for node gender decision

This method is slightly different than the predefined activity priority list method. In this method, node gender activity is selected as the one having most unallocated resources, if two activities equal to each other, the one with larger resource demand is selected. [1, 2, -] sequencing criteria is utilized. The priority list is updated dynamically as unallocated resource amount may change depending on reschedule. NGD3 has been used for the RID-MRD objective function.

5.5.4 Branching strategy

Branching is the process of generating new nodes from a candidate mother node. Branching process continues until optimal solution is found. As branching applied, newly generated

nodes are increased one level in generation and it can be applied to all nodes except leaves (end nodes). When current node is finished for branching, from which node to continue is a raising issue in branch and bound procedure. Zamani (2001) has proposed to continue branching from the node having minimum lower bound value among all open nodes. In this thesis study, this branching method is utilized. Branching is continued from the smallest lower bound value (best) node. This process requires finding the best node at each branching process. Linking the nodes tree wise and tracing these links to find the best node is a time consuming process. Also, as defined in section 5.1, because a node only knows the address of its mother node, tracing is possible only towards the root node. Instead of tracing on the solution tree, a candidate mother node queue is formed in which nodes that can be branched are arranged from the best node to the worst node. As stated in node definition, each node also knows the address of the next coming best node. Considering the queue, each node knows the address of its successor node. At each time, the node that will be imposed to generate new nodes is taken from the head of the queue. The next coming node becomes the head. During branching, newly generated nodes are placed into the queue to the correct places according to their lower bound values. This way, the node for branching can be picked easily and precisely. Dealing with the candidate mother node queue requires a special effort because for difficult problems its size extremely enlarges. This issue has been solved by an indexing method which is explained in the next section.

5.5.5 Candidate mother node queue and queue indexing

The candidate mother node queue is used to keep the nodes arranged for branching and to easily pick the best node for the next branching process. The head of the queue is the best node and the tail is the worst node. The worst node can have a lower bound value equal to the upper bound value at most since the nodes with greater lower bound value than the upper bound value are non-promising and they are just deleted. When the head node of the queue is imposed for branching, its link is broken from the queue and stays at memory just as a mother node. The next coming node becomes the head of the queue. Newly generated child nodes are checked if they are capable of new node generation (i.e. not leaf nodes) and if they are promising. Eligible child nodes are selected candidate mother nodes and placed to the queue according to their lower bound values. They are placed just in front of the nodes having the same lower bound value. Finding the correct place for a new node requires searching in the queue. Generally, implemented searching method starts from the head node and compares lower bound values until correct place is found. Nevertheless, for difficult problem instances, the size of the queue reaches to billion nodes. At such cases, for each new node placement, billions of nodes are searched. Every time, lower bound information of a node is looked up. If it is smaller than the new node value, the address of the next best node is traced. Looking up for data and comparing values are repeated until correct place of the new node is found. Considering that this bulky searching process is repeated for each newly generated node, as the size of the queue increases, the speed of the algorithm severely drops to plodding speeds.

The arrangement in the candidate mother node queue is performed according to the lower bound values. The smallest lower bound value is found just at the beginning of the tree

formation process and belongs to the root node. The biggest lower bound value is equal to the upper bound value. As a result, the range of the lower bound values in the queue is definite. All candidate mother nodes will have values falling in this range. A new node is placed just in front of the nodes with the same value in the queue. Considering these facts, it has been realized that if the address of the frontmost node of the same lower bound value is known, the new node can be placed directly. Therefore, addresses of frontmost nodes belonging to all lower bound values in the range are kept in memory as indexed. Once a new node is to be placed in the queue, the queue index is determined corresponding to its lower bound value. Without searching in the queue, the address of the correct frontmost node is found by the queue index and the new node is placed in front of that node. After placement finishes, necessary address updates are performed and next process is initiated. This method is queue size independent. By queue indexing the management of the candidate mother node queue is achieved with negligible effort even for difficult problem instances. The method has omitted all the drawbacks of link-wise searching.

Figure 5.12 illustrates a candidate mother node queue with queue indexing. The root node value is assumed to be 37, upper bound value is to be 50 and the optimal value is to be 48. Queue indexing is in the range of 37-50 which has 14 addresses of frontmost nodes. At the beginning of the solution tree, branching starts with the root node and the queue is empty. As new nodes are generated, the queue is filled. Newly generated child nodes having lower bound values greater than 50 (greater that the upper bound value) are deleted. Child nodes with values less than 50 are placed in the queue. The address at the queue index corresponding to the lower bound value is read and the new node is placed in front of the node group with the same value. The existing link is broken and relinking is performed with the new node. Each time, the head node is used for branching. Newly generated nodes cannot have a lower bound value smaller than that of the head node. This is guaranteed by the lower bound calculation strategy – child nodes cannot have better values than their mother node – and by processing already on the best node. As the branching process goes on, the head node value steps up towards the optimal value. After a certain time, through the branching process, some mother nodes generate leaves. In this case, the leaf that has a better value than the upper bound value (say 48) is taken as the current best solution. Upper bound value is updated. Candidate mother nodes at index 48 and above lose their capability of producing better solutions. Therefore they are never branched from. Whenever no candidate mother node remains with a value under 48, the current best solution is reported as the optimal solution to the problem. All nodes below the optimal value should be branched with the priority for branching not being important. However, as the optimal value is not known during the procedure, arrangement of the nodes from best to the worst plays a crucial role not to branch the nodes between the optimal and initial upper bound value. Some queue indexes may remain empty for the whole branching process if a node with the corresponding lower bound value does not exist. The proposed queue indexing method significantly improved the computation performance compared to link-wise searching method. For most of the problems, almost 99% of the computation time was saved.

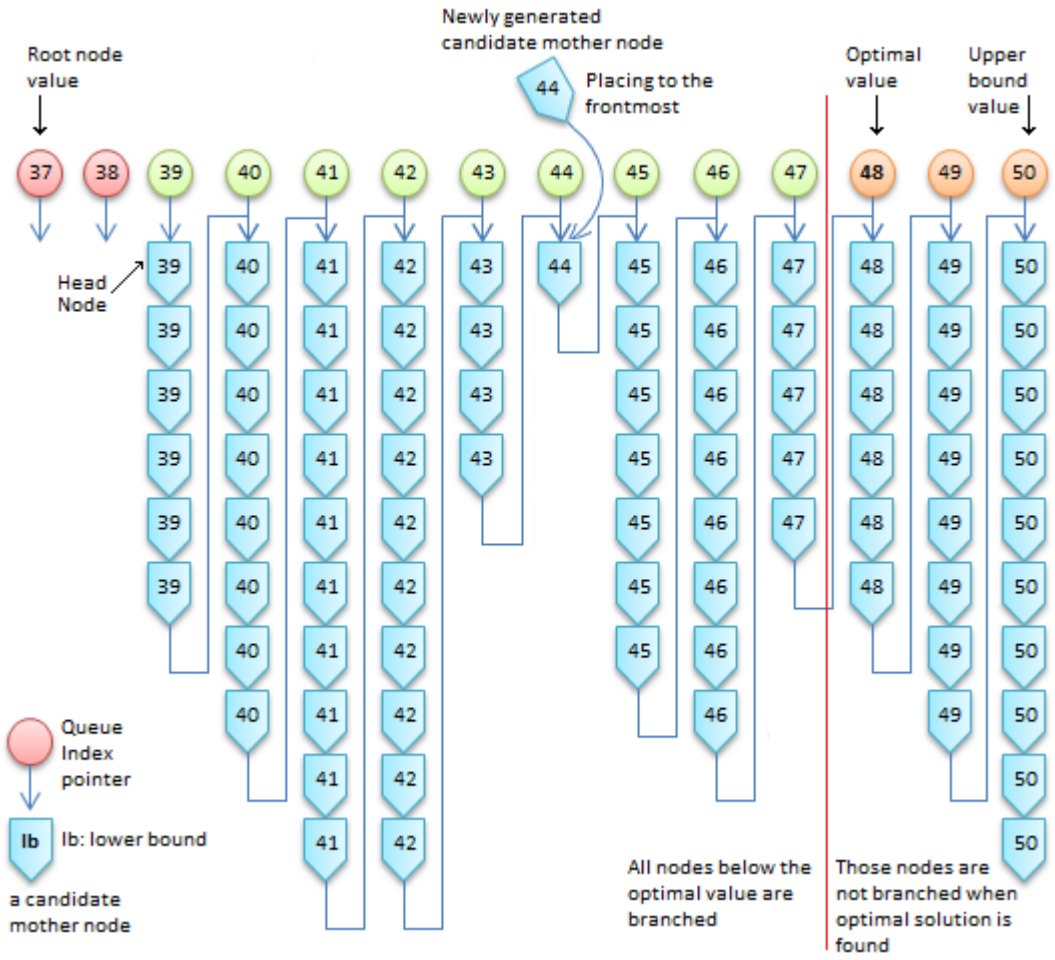


Figure 5.12 Candidate mother node queue with queue indexing

5.6 Summary of the Proposed Branch and Bound Procedure and Process Overview

The proposed branch and bound algorithm has many simple and effective techniques integrated within to increase its capability to solve larger problems and improve its performance. An upper bound value is determined by the adapted branch and bound heuristic. The root node is formed and the smallest lower bound value is also determined. The candidate mother node queue indexing size is defined in the range of the smallest lower bound value and the upper bound value. Branching process is initiated by generating new nodes from the root node which is also the only node in the queue. For the rest of branching processes, the head node of the candidate mother node queue is used. By tracing the ancestor nodes of the mother node schedule map is obtained. Using schedule map, rescheduling is performed to update early and late start times. The node gender for child nodes is determined either using predefined activity priority list or using average resource effect method. Once the node gender is defined, new nodes are generated. For each newly generated node, two lower bound values are calculated using LBC-1 and LBC-2 strategies.

The tighter (greater) value is coded to the node. Nodes are checked for being promising. Non-promising nodes are deleted. Queue index is determined corresponding to the lower bound value for the eligible nodes. Newly generated eligible nodes are placed in the queue. The head node of the queue is occupied for branching and removed from the queue. As branching continues, more new nodes are generated, the size of the queue increases, and the average generation level of the candidate mother nodes increases. Towards the end of the solution tree, last generation level nodes come to the head of the queue and newly generated nodes become leaves. Each leaf is compared with the upper bound value as it contains a complete schedule. Upper bound value is updated if a better value is obtained. Candidate mother nodes that fall above the new upper bound value due to the update are not eligible for branching any more. However, deletion of them implies a link-wise search and delete process, which may require tedious effort. Thus, they are left in memory as garbage. Among the generated leaves, only the best leaf is kept in memory and remaining is deleted. The leaf is also checked for optimality. This is achieved by looking for existence of eligible nodes in the queue. After the upper bound update, if eligible candidate mother nodes exist in the queue, branching process is continued. However, if no more eligible candidate mother node (no head node) remains in the queue, the solution process is terminated and the scheduled map obtained from the best leaf is reported as the optimal solution. Figure 5.13, the flow chart of the developed algorithm for solving RLP to optimality has been shown.

When the developed branch and bound algorithm is analyzed in terms of memory use, it can be seen that two main groups occupy memory. The first group is the node pile that falls in the region of optimal value and upper bound value. Those nodes remain in memory as their deletion is a time consuming process. They can be eliminated not to occur as long as the initial upper bound is equal to the optimal solution. The second group is the mother node population. All of the candidate mother nodes are kept in memory even after they are used for new node generation. They are necessary at later times for obtaining schedule map as node data contains the start date of only the node gender activity. Each node might be assigned all start dates of scheduled activities, however, that increases the individual size of node and reduces the capacity performance as well. On the other hand, considering the computation time performance of the procedure, the functions that are called frequently are simplified as much as possible. Also, the quality of the lower bound calculation is important in reducing the number of branching applied nodes. However, the more complexity is introduced to the algorithm, the more computation is required. Despite the limitations, branch and bound algorithm still can be improved and developed further to increase its performance and solving capability for larger size problems.

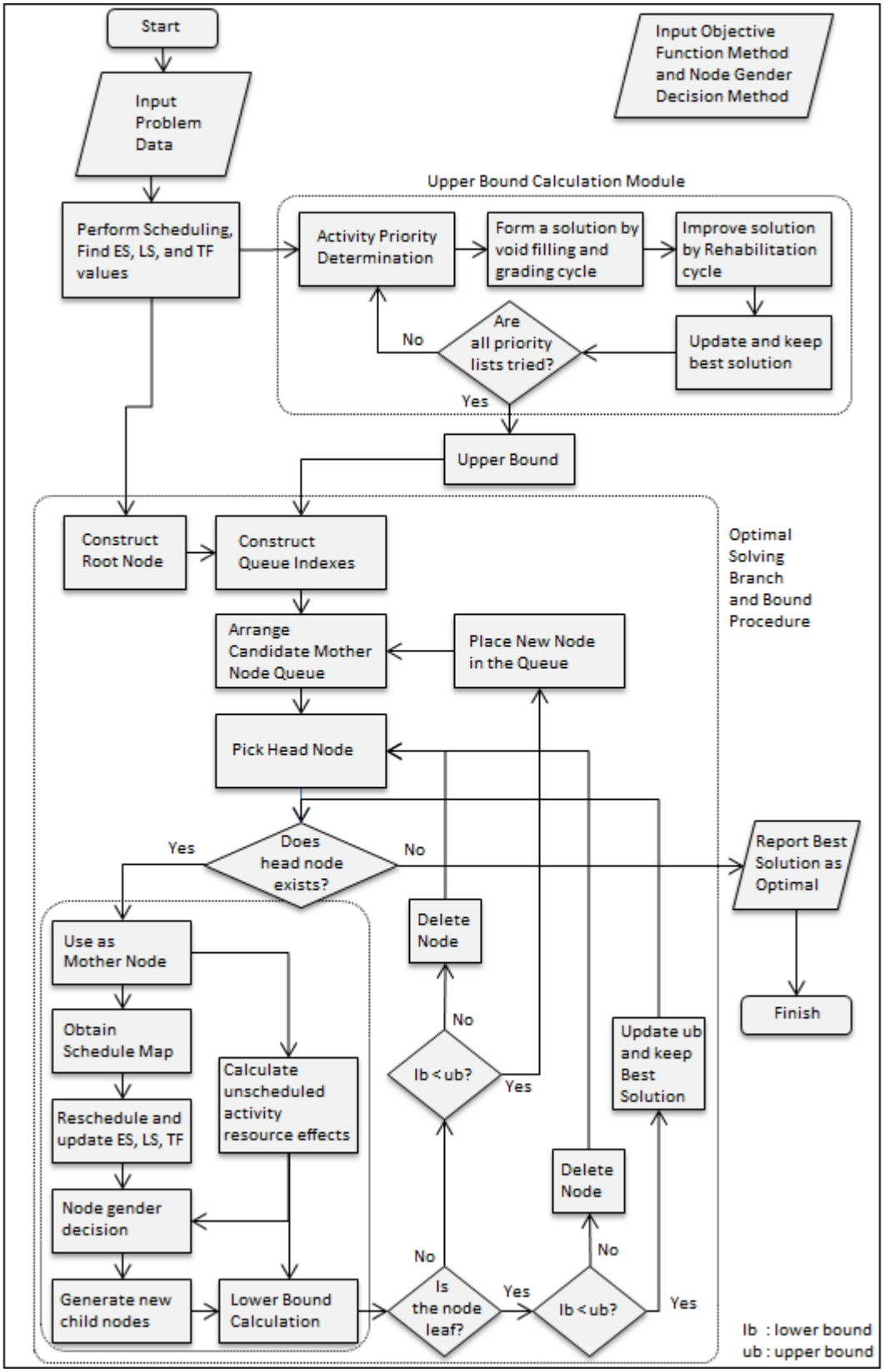


Figure 5.13 Flow chart of the developed RLP optimal solving algorithm

CHAPTER 6

MIXED-INTEGER LINEAR MODEL FOR RLP

In addition to branch and bound algorithm, other methods also have been widely used in the literature. One of the most popular methods is integer linear programming. The recent research study published by Rieck et al. (2012) uses a mixed-integer linear optimization model for the resource leveling problem. However, experimentations have been carried out for two objective functions; minimization of sum of squares and minimization of overload resources. The proposed branch and bound algorithm is tested using the same test set of Rieck et al. (2012). One of the aims of this thesis study is to develop and solve RLP using RID-MRD metric. Therefore, using mixed-integer linear programming a model is developed. Branch and bound algorithm and mixed-integer linear model have been compared for solving RLP based on RID-MRD. Results have been given in Chapter 7. The mixed-integer linear model is integrated into MS Excel and GAMS/CPLEX software to simplify application. In the following, details of model have been presented.

6.1 Definitions of Model Inputs

6.1.1 Sets

I: Project activities, $i = 1, \dots, I$.

T: Time intervals within the project duration, $t = 0, \dots, T$.

K: Resource types used in the project, $k = 1, \dots, K$.

N: Daily total demand by all activities from a resource type, $n = 1, \dots, N$.

6.1.2 Parameters

EST_i : Early start time of activity i.

LST_i : Late start time of activity i.

d_i : Duration of activity i.

$r_{i,k}$: Resource demand of activity i from resource type k.

$a_{t,k}$: The targeted resource demand at time t from resource type k.

w_k : Weight of resource type k.

D : Project deadline.

$p_{i,j}$: $\begin{cases} 1 & \text{if activity } j \text{ should be finished before activity } i; \\ 0 & else. \end{cases}$

6.1.3 Variables

z_1 : Weighted sum of square of daily resource demands of all resource types.

z_2 : Weighted sum of absolute deviations of daily resource demands from the targeted resource demand of all resource types.

z_3 : Weighted sum of maximum daily resource demand of all resource types.

z_4 : Weighted sum of resource idle days of all resource types.

f_i : Start day of activity *i*.

$u_{t,k}$: Daily demand of resource k at time t.

mxu_k : Maximum daily demand of resource k.

$mx1u_{t,k}$: Maximum daily demand of resource k before time t.

$mx2u_{t,k}$: Maximum daily demand of resource k after time t.

$mnu_{t,k}$: The smallest of *mx1* or *mx2* for each time interval and for each resource type.

$v_{t,k}$: Square of daily demand of resource k at time t.

$x_{t,k}$: Excess demand from the targeted resource level ($a_{t,k}$) of resource k at time t where over demand periods occur.

$y_{t,k}$: Less demand from the targeted resource level ($a_{t,k}$) of resource k at time t where under demand periods occur.

$\lambda_{n,t,k}$: $\begin{cases} 1 & \text{if demand for resource } k \text{ at time } t \text{ is equal to } n; \\ 0 & else. \end{cases}$

$\varphi_{t,i}$: $\begin{cases} 1 & \text{if activity } i \text{ is under progress at time } t; \\ 0 & else. \end{cases}$

$\sigma_{t,i}$: $\begin{cases} 1 & \text{if activity } i \text{ has started at time } t; \\ 0 & else. \end{cases}$

6.2 Modeling

The resource leveling objective functions that have been mentioned in Chapter 3 have been modeled based on mixed-integer linear programming. Equation (6.1) tries to minimize the weighted sum of squares of daily resource demands. Equation (6.2) tries to minimize the weighted sum of absolute deviations of daily resource demands from the targeted resource level. Equation (6.3) tries to minimize the weighted sum of excess resource usage over the targeted resource level. Equation (6.4) – which is MRD (maximum resource demand) – minimizes the weighted sum of peak demands of each resource type. Equation (6.5) – which is RID (resource idle days) – tries to minimize the weighted sum of resource idle days occurring at each resource utilization profile.

$$\min \quad z_1 = \sum_t \sum_k w_k {u_{t,k}}^2 \tag{6.1}$$

$$\min \quad z_2 = \sum_t \sum_k w_k |u_{t,k} - a_{t,k}| \tag{6.2}$$

$$\min \quad z_3 = \sum_t \sum_k w_k (u_{t,k} - a_{t,k})^+ \tag{6.3}$$

$$\min \quad z_4 = \sum_k w_k max(u_{1,k}, \dots, u_{T,k}) \tag{6.4}$$

$$\min \quad z_5 = \sum_t \sum_k w_k (min(max(u_{1,k}, \dots, u_{t,k}), max(u_{t,k}, \dots, u_{T,k})) - u_{t,k}) \tag{6.5}$$

Because above mentioned functions are not linear, they are first linearized. In the following sections, each linear representation of the function is given with its constraints different than the others, and then, common constraints shared by all of the models have been stated.

6.2.1 Model for sum of squares metric

$$\min \quad z_1 = \sum_t \sum_k w_k \, v_{t,k} \tag{6.6}$$

Constraints:

$$u_{t,k} = \sum_n n\lambda_{n,t,k} \quad \forall t \in T, \forall k \in K \tag{6.7}$$

$$v_{t,k} = \sum_n n^2 \lambda_{n,t,k} \quad \forall t \in T, \forall k \in K \tag{6.8}$$

$$\sum_n \lambda_{n,t,k} = 1 \quad \forall t \in T, \forall k \in K \tag{6.9}$$

$$v_{t,k} \in Z_0 \quad \forall t \in T, \forall k \in K \tag{6.10}$$

$$\lambda_{n,t,k} \in \{0,1\} \quad \forall n \in N, \forall t \in T, \forall k \in K \tag{6.11}$$

The model (6.6) aims to minimize the weighted sum of squares of daily resource demands. Equation (6.7) finds the total use of resource *k* at time *t*. Equation (6.8) is equal to the square of that value. Equation (6.9) imposes the daily total use of resource *k* to take a single value. $v_{t,k}$ variables can take the value of zero or positive integer values. $\lambda_{n,t,k}$ are binary variables.

6.2.2 Model for absolute deviations metric

$$\min \quad z_2 = \sum_t \sum_k w_k (x_{t,k} + y_{t,k}) \tag{6.12}$$

Constraints:

$$u_{t,k} - a_{t,k} = x_{t,k} - y_{t,k} \quad \forall t \in T, \forall k \in K \tag{6.13}$$

$$x_{t,k}, y_{t,k} \in Z_0 \quad \forall t \in T, \forall k \in K \tag{6.14}$$

The model (6.12) aims to minimize the weighted sum of absolute deviations of daily resource demands from the targeted resource level. Due to the nonlinearity of the absolute value function, the constraint (6.13) ensures linearity of the model by implying $u_{t,k} - a_{t,k}$ as a difference of two non-negative integer variables. $x_{t,k}, y_{t,k}$ variables can take the value of zero or positive integer values.

6.2.3 Model for overload resources metric

$$\min \quad z_3 = \sum_t \sum_k w_k x_{t,k} \tag{6.15}$$

Constraints:

$$x_{t,k} \in Z_0 \quad \forall t \in T, \forall k \in K \tag{6.16}$$

The model (6.15) aims to minimize the weighted sum of overload resources from the targeted resource level. $x_{t,k}$ variable can take the value of zero or positive integer values.

6.2.4 Model for maximum resource demand metric

$$\min \quad z_3 = \sum_k w_k mxu_k \tag{6.17}$$

Constraints:

$$u_{t,k} \leq mxu_k \quad \forall t \in T, \forall k \in K \tag{6.18}$$

$$mxu_k \in Z_0 \quad \forall k \in K \tag{6.19}$$

The model (6.17) aims to minimize the weighted sum of peak demands of each resource type. mxu_k is a variable that represents maximum daily usage of resource type *k*. After that definition, daily resource usages of each resource is restricted not to exceed corresponding

mxu_k value. With this constraint (6.18) the model is linearized. mxu_k variables can take the value of zero or positive integer values.

6.2.5 Model for resource idle days metric

$$\min \quad z_4 = \sum_t \sum_k w_k (mnu_{t,k} - u_{t,k}) \tag{6.20}$$

Constraints:

$$u_{t',k} \leq mx1u_{t,k} \quad \forall t' \in t, \forall t \in T, \forall k \in K \tag{6.21}$$

$$u_{t',k} \leq mx2u_{t,k} \quad \forall t' \in t, \forall t \in T, \forall k \in K \tag{6.22}$$

$$mnu_{t,k} \leq mx1u_{t,k} \quad \forall t \in T, \forall k \in K \tag{6.23}$$

$$mnu_{t,k} \leq mx2u_{t,k} \quad \forall t \in T, \forall k \in K \tag{6.24}$$

$$mx1u_{t,k}, mx2u_{t,k}, mnu_{t,k} \in Z_0 \quad \forall t \in T, \forall k \in K \tag{6.25}$$

The model (6.20) aims to minimize the weighted sum of resource idle days. In other words, it tries to minimize the weighted sum of differences between the resource demand of each day and the smallest one of the largest resource demands before and after that day. $mx1u_{t,k}$ is the largest demand of resource *k* before time *t*. On the other hand, $mx2u_{t,k}$ is the largest demand of resource *k* after time *t*. Constraints (6.21) and (6.22) imply that daily demands of resource *k* before and after time *t* cannot be greater than the largest demands of resource *k* before and after time *t*. Inequalities (6.23) and (6.24) finds the smallest one of the largest daily demands of resource *k* before and after time *t* for each day and each resource type. $mx1u_{t,k}$,$mx2u_{t,k}$ $mnu_{t,k}$ variables can take the value of zero or positive integer values.

6.2.5 Common constraints

$$\sum_i r_{i,k} \varphi_{t,i} = u_{t,k} \quad \forall t \in T, \forall k \in K \tag{6.26}$$

$$p_{i,j} f_i \geq p_{i,j} \left(f_j + d_j \right) \quad \forall i, j \in I, i \neq j \tag{6.27}$$

$$\sum_{EST_i \leq t \leq LST_i} t\sigma_{t,i} = f_i \quad \forall i \in I \tag{6.28}$$

$$\sum_{EST_i \leq t \leq LST_i} \sigma_{t,i} = 1 \quad \forall i \in I \tag{6.29}$$

$$\varphi_{t,i} = \sum_{t=\max(EST_i, t-d_i+1)}^{\min(LST_i, t)} \sigma_{t,i} \quad \forall t \in T, \forall i \in I, EST_i \leq t \leq LST_i + d_i - 1 \tag{6.30}$$

$$\varphi_{t,i} = 0 \quad \forall t \in T, \forall i \in I, t < EST_i \tag{6.31}$$

$$\varphi_{t,i} = 0 \quad \forall t \in T, \forall i \in I, t > LST_i + d_i - 1 \tag{6.32}$$

$$f_1 = 0 \tag{6.33}$$

$$f_I \leq D \tag{6.34}$$

$$\sigma_{0,1} = 1 \tag{6.35}$$

$$u_{t,k} \in \aleph_o \quad \forall t \in T, \forall k \in K \tag{6.36}$$

$$f_i \in Z_o \quad \forall i \in I \tag{6.37}$$

$$\varphi_{t,i} \in \{0,1\} \, \forall t \in T, \forall i \in I \tag{6.38}$$

$$\sigma_{t,i} \in \{0,1\} \, \forall t \in T, \forall i \in I \tag{6.39}$$

All constraints related to the scheduling of the activities are common for all models. Equation (6.26) finds the daily resource demand. The activities are constricted to use resources only within the time they are in progress with this constraint. Constraint (6.27) prevents starting of an activity unless its predecessors are finished. (6.28) finds the starting time of activities. Constraint (6.29) implies that each activity can be started in a time only in between its early start and late start times. Equation (6.30) determines the period of which the activity is on progress and it ensures the continuity of this period. Constraints (6.31) and (6.32) imply that an activity cannot be active on a time before its early start time and after its late finish time. (6.33) states that the dummy start activity starts at time zero and (6.34) states that the dummy finish activity ends before time D implying the project start time and due times. (6.35) denotes that the start activity is active at time zero. $u_{t,k}$; f_i variables can take the value zero or positive integer values. $\varphi_{t,i}, \sigma_{t,i}$ variables are binary variables.

The mixed-integer linear model has been implemented into an MS Excel file. Instance input data is entered using a sheet prepared in MS Excel. The problem data is prepared according to the model by the Excel file, and then, computational processes are performed by GAMS/CPLEX software. Results are printed in a sheet of the Excel file. Experimentation results have been presented in Chapter 7.

CHAPTER 7

COMPUTATIONAL EXPERIMENT RESULTS

The developed branch and bound algorithm is capable of solving resource leveling problem subjected to precedence relations and general temporal constraints. One of the four objective functions stated in Chapter 3 can be utilized to measure leveling. For the objective functions except RID-MRD, LBC-1 and LBC-2 strategies have been utilized together for lower bound calculation, two node gender decision methods (NGD1 and NGD2) have been implemented and test results corresponding to both methods have been presented. For RID-MRD metric, only LBC-1 has been employed. LBC-2 is not suitable for this metric, hence, it is not applied. For node gender decision, NGD3 method has been utilized for RID-MRD.

Testing of the algorithm has been performed on a computer with Intel® Core™ i5-2500 CPU at 3.30 GHz and an operating system of 64 bit. 16 GB RAM capacity with a 120 GB SSD added as virtual memory, in total a net 127 GB computation capacity has been provided. Although, the employed computer has a processor speed of 3.30 GHz, time results are given scaled with a factor to enable comparison with some other experimentation results. Therefore, for comparison with time results of Gather et al. (2010) and Rieck et al. (2012) the factor is 1.24 (3.30/2.67) to comply with 2.67 GHz processing speed.

Computational study has been conducted using the following test sets. A 21 problem set with sizes 5 to 20 activities collected from the literature has been solved using branch and bound algorithm based on SSQR, ABSDEV, RID-MRD objective functions (Mutlu, 2010). Results are given in section 7.1 in comparison with a start time enumeration based branch and bound procedure of Mutlu (2010). A test set of 48 problems with sizes of 30 activities selected from PSPLIB has been solved using branch and bound algorithm and mixed-integer linear model based on RID-MRD objective function. Results have been given in section 7.2. Test sets *rlp_j10*, *rlp_j20* of Kolisch et al. (1999) and test sets *ubo10*, *ubo20* of Franck et al. (2001) have been solved using branch and bound algorithm based on SSQR objective function. Results have been given in section 7.3 in comparison with Gather et al. (2010). Test set T_2 of *rlp_10*, *rlp_15*, *rlp_20* and *rlp_30* of 1, 3, and 5 resource cases of Rieck et al. (2012) have been solved using branch and bound algorithm and mixed-integer linear model based on SSQR, OVERLOAD and RID-MRD objective functions. Results have been given in section 7.4 in comparison with Rieck et al. (2012).

7.1 Computational Results of 21 Problem Test Set

A 21 problem test set collected from the literature consisting of instances with sizes 5 to 20 activities with single and multi-resource cases have been solved using branch and bound algorithm (Mutlu, 2010). The instances in the test set are subjected to precedence relations. The test set has been solved utilizing SSQR, ABSDEV and RID-MRD objective functions. The same test set has also been solved by start time enumeration based branch and bound algorithm developed by Mutlu (2010) according to Zimmermann (2000). Both tests have been performed on the same computer with 3.30 GHz processor speed. The same optimal results with the results of Mutlu (2010) have been obtained. Table 7.1, 7.2 and 7.3 represent the computation time results. Optimal results of the problem set are available in Appendix A. Optimal resource profiles of the problem of El-Rayes and Jun (2009) are available in Appendix B.

Table 7.1 Computation time results of the 21 problem set according to SSQR metric

SSQR (weighted sum of squares of daily resource demands)			Computational Time (s)			
			This Study (B&B with NGD2)		Start Time Enumeration Based Branch and Bound (Mutlu, 2010)	
	Problem	Act.	Single Resource	Multi Resource	Single Resource	Multi Resource
1	Demeulemeester (2002 Pg.416)	10	0	0.015	0	0
2	Easa (1989)	5	0	0	0	0
3	El Rayes and Jun (2009)	20	0.016	0.015	1	30
4	Generated II (Mutlu 2010)	6	0	0	0	0
5	Generated III (Mutlu 2010)	14	0	0.016	0	0
6	Generated IV (Mutlu 2010)	18	0	0.015	0	0
7	Generated V (Mutlu 2010)	13	0	0	0	0
8	Generated VI (Mutlu 2010)	14	0.016	0.015	0	6
9	Generated VII (Mutlu 2010)	16	0.016	0.015	2	3
10	Harris (1990)	11	0	0.016	0	0
11	Hinze (2004 Pg.152)	15	0	0.016	68	53
12	Leu (2000)	13	0.015	0.093	30	43
13	Mubarak (2004 Pg.61)	14	0	0.016	7	0
14	Mubarak (2004 Pg.67)	11	0	0	0	0
15	Mubarak (2004 Pg.217)	8	0	0	0	0
16	Newitt (2005 Pg.82)	16	0	0.016	0	0
17	Newitt (2005 Pg.121)	12	0	0.016	0	0
18	Son and Skibniewski (1999)	13	0	0.015	1	0
19	Son and Skibniewski (1999)	11	0	0.016	0	3
20	Stevens (1990 Pg.97)	15	0.016	0.11	72	529
21	Stevens (1990 Pg.172)	19	0.031	0.046	10	104
	Average Time		0.005	0.021	9.095	36.714

Table 7.2 Computation time results of the 21 problem set according to ABSDEV metric

ABSDEV (weighted sum of absolute deviations from the average resource demand)			Computational Time (s)			
			This Study (B&B with NGD2)		Start Time Enumeration Based Branch and Bound (Mutlu, 2010)	
	Problem	Act.	Single Resource	Multi Resource	Single Resource	Multi Resource
1	Demeulemeester (2002 Pg.416)	10	0.015	0.015	0	0
2	Easa (1989)	5	0	0	0	0
3	El Rayes and Jun (2009)	20	0.016	0.032	2	48
4	Generated II (Mutlu 2010)	6	0	0.015	0	0
5	Generated III (Mutlu 2010)	14	0	0.016	0	0
6	Generated IV (Mutlu 2010)	18	0	0.015	0	0
7	Generated V (Mutlu 2010)	13	0	0.016	0	0
8	Generated VI (Mutlu 2010)	14	0	0.031	0	8
9	Generated VII (Mutlu 2010)	16	0	0.031	2	6
10	Harris (1990)	11	0	0.016	0	0
11	Hinze (2004 Pg.152)	15	0	0.015	69	90
12	Leu (2000)	13	0.031	0.078	36	54
13	Mubarak (2004 Pg.61)	14	0.016	0.015	10	38
14	Mubarak (2004 Pg.67)	11	0.015	0	0	0
15	Mubarak (2004 Pg.217)	8	0.016	0	0	0
16	Newitt (2005 Pg.82)	16	0	0.016	0	1
17	Newitt (2005 Pg.121)	12	0	0	0	0
18	Son and Skibniewski (1999)	13	0	0.015	1	4
19	Son and Skibniewski (1999)	11	0	0	0	0
20	Stevens (1990 Pg.97)	15	0.016	0.187	63	728
21	Stevens (1990 Pg.172)	19	0.031	0.046	16	158
	Average Time		0.007	0.027	9.476	54.048

Absolute deviations have been based on average resource demand. The average calculations have been carried out using standard rounding. Weights of RID and MRD have been equally distributed and assigned as 1 for each metric.

Computational results based on minimization of sum of squares, absolute deviation and resource idle day with maximum daily resource demand methods show that the developed branch and bound algorithm with the applied techniques has performed far better than the start time enumeration based branch and bound algorithm proposed by Zimmermann (2000) and developed by Mutlu (2010). Average computation time is less than 0.1 seconds for both single and multi-resource cases and all objective functions for the developed branch and bound algorithm. However, it is about tens of seconds for the start time enumeration based branch and bound procedure. Besides, the average computation time of the RID-MRD solutions for the multi resource case reaches about 500 seconds.

Table 7.3 Computation time results of the 21 problem set according to RID-MRD metric

	RID-MRD (weighted sum of (1*resource idle days) + (1*maximum daily resource demand))		Computational Time (s)			
			This Study (B&B with NGD2)		Start Time Enumeration Based Branch and Bound (Mutlu, 2010)	
	Problem	Act.	Single Resource	Multi Resource	Single Resource	Multi Resource
1	Demeulemeester (2002 Pg.416)	10	0.015	0	0	0
2	Easa (1989)	5	0.016	0	0	0
3	El Rayes and Jun (2009)	20	0.015	0.047	39	3414
4	Generated II (Mutlu 2010)	6	0	0	0	0
5	Generated III (Mutlu 2010)	14	0.015	0.016	0	0
6	Generated IV (Mutlu 2010)	18	0	0.016	0	0
7	Generated V (Mutlu 2010)	13	0	0.016	1	2
8	Generated VI (Mutlu 2010)	14	0	0.047	3	86
9	Generated VII (Mutlu 2010)	16	0.015	0.046	49	228
10	Harris (1990)	11	0	0	0	0
11	Hinze (2004 Pg.152)	15	0	0	17	45
12	Leu (2000)	13	0.015	0.141	216	261
13	Mubarak (2004 Pg.61)	14	0.015	0.062	72	835
14	Mubarak (2004 Pg.67)	11	0	0.016	0	0
15	Mubarak (2004 Pg.217)	8	0	0	0	0
16	Newitt (2005 Pg.82)	16	0.016	0.078	5	17
17	Newitt (2005 Pg.121)	12	0	0.015	0	0
18	Son and Skibniewski (1999)	13	0.015	0.016	28	144
19	Son and Skibniewski (1999)	11	0.016	0	0	1
20	Stevens (1990 Pg.97)	15	0.015	0.53	493	4009
21	Stevens (1990 Pg.172)	19	0.031	0.187	257	854
	Average Time (s)		0.009	0.059	56.190	471.238

7.2 Computational Results of 48 Problem Set from PSPLIB

A problem set of 48 problems with sizes of 30 activities from the PSPLIB of Kolisch et al. (1997) has been solved using branch and bound algorithm and mixed-integer linear model based on RID-MRD objective function. The problems are subjected to precedence relations. Computations for the developed branch and bound algorithm have been carried out using the 3.30 GHz computer. The mixed-integer linear model has been run using a 2.67 GHz computer. Time results of branch and bound algorithm are scaled and all time results are given based on 2.67 GHz processing speed. Each algorithm has been given a computation time limit of 3 hours. The developed branch and bound algorithm has been able to solve 28 of the 48 problems to optimality, whereas mixed-integer linear model has been able to solve only 5 of the 48 problems within the specified computation time limit. Table 7.4 presents the timing results. It has been shown that the developed branch and bound algorithm outperforms the mixed-integer linear modeling for solving RLP based on RID-MRD metric.

Table 7.4 Computation time results of 48 problem set from PSPLIB (Kolisch et al., 1997) based on RID-MRD metric (1/2)

	RID-MRD (weighted sum of (1*resource idle days) + (1*maximum daily resource demand))		Computational Time (s)			
			This Study (B&B with NGD3)		This Study (Mixed-Integer Model 4&5)	
	Problem	Optimal	Time (s)	3hr Limit	Time (s)	3hr Limit
1	J301_1.RCP	89		-	170	✓
2	J302_1.RCP	229	389.029	✓	257	✓
3	J303_1.RCP	556	103.575	✓		-
4	J304_1.RCP	237	8,619.200	✓		-
5	J305_1.RCP	243	153.095	✓		-
6	J306_1.RCP	201	3,633.613	✓		-
7	J307_1.RCP	--		-		-
8	J308_1.RCP	307	21.869	✓		-
9	J309_1.RCP	--		-		-
10	J3010_1.RCP	236	408.801	✓		-
11	J3011_1.RCP	--		-		-
12	J3012_1.RCP	--		-		-
13	J3013_1.RCP	223	913.861	✓		-
14	J3014_1.RCP	234	41.243	✓		-
15	J3015_1.RCP	--		-		-
16	J3016_1.RCP	--		-		-
17	J3017_1.RCP	--		-		-
18	J3018_1.RCP	462	743.973	✓		-
19	J3019_1.RCP	211	23.325	✓		-
20	J3020_1.RCP	629	49.635	✓	3850	✓
21	J3021_1.RCP	581	610.878	✓	200	✓
22	J3022_1.RCP	390	15.560	✓		-
23	J3023_1.RCP	658	2,520.088	✓		-
24	J3024_1.RCP	261	997.155	✓		-
25	J3025_1.RCP	--		-		-
26	J3026_1.RCP	--		-		-
27	J3027_1.RCP	232	467.286	✓		-
28	J3028_1.RCP	--		-		-
29	J3029_1.RCP	--		-		-
30	J3030_1.RCP	--		-		-
31	J3031_1.RCP	--		-		-
32	J3032_1.RCP	--		-		-
33	J3033_1.RCP	531	4.956	✓		-
34	J3034_1.RCP	--		-		-
35	J3035_1.RCP	--		-		-
36	J3036_1.RCP	--		-		-
37	J3037_1.RCP	354	370.481	✓		-
38	J3038_1.RCP	339	48.594	✓		-
39	J3039_1.RCP	357	3,820.863	✓		-
40	J3040_1.RCP	401	3,391.900	✓		-
41	J3041_1.RCP	761	0.106	✓	35	✓

Table 7.4 Computation time results of 48 problem set from PSPLIB (Kolisch et al., 1997) based on RID-MRD metric (2/2)

	RID-MRD (weighted sum of (1*resource idle days) + (1*maximum daily resource demand))		Computational Time (s)			
			This Study (B&B with NGD3)		This Study (Mixed-Integer Model 4&5)	
	Problem	Optimal	Time (s)	3hr Limit	Time (s)	3hr Limit
42	J3042_1.RCP	--		-		-
43	J3043_1.RCP	476	107.545	✓		-
44	J3044_1.RCP	248	59.065	✓		-
45	J3045_1.RCP	--		-		-
46	J3046_1.RCP	296	3,455.500	✓		-
47	J3047_1.RCP	460	4,761.025	✓		-
48	J3048_1.RCP	432	6,268.413	✓		-
	Average Time (s)		1,500.023	28/48	902.400	5/48

7.3 Computation Results of Test Sets *rlp_j10, rlp_j20* of Kolisch et al. (1999) and Test Sets *ubo10, ubo20* of Franck et al. (2001)

Test sets *rlp_j10*, *rlp_j20* of Kolisch et al. (1999) and test sets *ubo10*, *ubo20* of Franck et al. (2001) have been solved based on sum of squares metric using the developed branch and bound algorithm with the 3.30 GHz computer. The problem instances are subjected to general temporal constraints. Average resource effect based node gender decision method (NGD2) has been utilized for the experimentation. The same test sets have also been experimented by Gather et al. (2010) using bridge-based approach, tree adaption and start time enumeration methods based on SSQR objective function. The computational results of Gather et al. (2010) have been based on 2.67 GHz processing speed. The optimal results of sets *rlp_j10* and *rlp_j20* were validated by checking the optimal results of Gather et al. (2010). Optimal results of *ubo10* and *ubo20* test sets of Gather et al. (2010) could have not been accessed. Computation time results of the developed branch and bound algorithm have been scaled and all results have been given based on 2.67 GHz processing speed. In Table 7.5, 7.6, 7.7 and 7.8, the average computation times and number of instances solved in less than a duration of 1, 10, 1000, and 36000 seconds have been presented.

Table 7.5 Computation time results for *rlp_j10* test set (270 test instances)

	Average	# < 1 s	# < 10 s
This Study (B&B with NGD2)	0.017 s	270	270
Bridge-Based Approach	0.020 s	270	270
Tree Adaption	0.440 s	215	268
Start Time Enumeration	0.660 s	241	263

Table 7.6 Computation time results for *ubo10* test set (90 test instances)

	Average	# < 1 s	# < 10 s
This Study (B&B with NGD2)	0.034 s	90	90
Bridge-Based Approach	0.190 s	86	90
Tree Adaption	5.290 s	43	80
Start Time Enumeration	0.620 s	78	89

Table 7.7 Computation time results for *rlp_j20* test set (90 test instances)

	Average	# < 10 s	# < 1000 s	# < 36000 s
This Study (B&B with NGD2)	22.868 s	86	89	90
Bridge-Based Approach	373.380 s	35	85	90
Tree Adaption	6,069.970 s	15	41	90

Table 7.8 Computation time results for *ubo20* test set (90 test instances)

	Average	# < 10 s	# < 1000 s	# < 36000 s
This Study (B&B with NGD2)	28.832 s	76	90	90
Bridge-Based Approach	22.280 s	27	53	74
Tree Adaption	5,549.420 s	6	16	41

Results have shown that the developed branch and bound algorithm has outperformed the bridge-based approach, tree adaption and start time enumeration methods significantly. All problem instances have been able to be solved to optimality. Although the average computation time of branch and bound procedure seems greater than the bridge based approach in Table 7.8, the number of solved instances is more and instances that could not have been solved are not accounted at average time calculation.

7.4 Computation Results of Test Sets T_2 of Rieck et al. (2012)

The most extensive experimentation has been performed on the test set T_2 of Rieck et al. (2012). The test set has been generated using the ProGen/max problem instance generator (Schwindt, 1998). Problem instances are subjected to general temporal constraints. The utilized test set is composed of test groups of problems with 10, 15, 20, 30 and 50 activities and 1, 3 and 5 resources. The test sets of problems with 50 activities have not been used for testing as the computation time exceeded 3 hours time limit in general. Each group contains 40 instances, in total 480 problem instances have been experimented. This test set has been solved using branch and bound algorithm and mixed-integer linear model based on SSQR, OVERLOAD and RID-MRD objective functions. For the SSQR and OVERLOAD

functions, both predefined priority list and average resource effect methods for node gender decision system has been experimented separately. Rieck et al, (2012) has also used the same set to test their mixed-integer linear based algorithm with a 2.67 GHz computer. The test results have been given based on SSQR and OVERLOAD functions. Therefore, comparison of RID-MRD function has been done between the developed branch and bound algorithm and the mixed-integer linear model developed through this thesis study. The results of the proposed branch and bound algorithm and the mixed-integer linear model on solving the test set T_2 of Rieck et al. (2012) based on sum of squares and overload objective function has been validated by obtaining the same optimal results with Rieck et al. (2012). For the RID-MRD objective function, the same optimal results have been obtained by the branch and bound algorithm and the mixed-integer linear model. Computations of branch and bound algorithm and mixed-integer linear model have been performed based on the 3.30 GHz processing speed. However, those time results have been scaled and all results have been presented based on 2.67 GHz processing speed to simplify the interpretation. A computation time limit of 3 hours has been used. Results are shown in Tables 7.9, 7.10, 7.11 and 7.12. The branch and bound algorithm with node decision system of predefined priority list is denoted as B&B with NGD1, with the average resource effect is denoted as B&B with NGD2, and with the dynamic priority list is denoted as B&B with NGD3.

Table 7.9 Computation results for *rlp_10* test sets

		rlp_10_1		rlp_10_3		rlp_10_5	
	Method	Time (s)	✓	Time (s)	✓	Time (s)	✓
SSQR							
This Study	B&B with NGD1	0.012	40	0.013	40	0.021	40
	B&B with NGD2	0.011	40	0.014	40	0.024	40
	Mixed-Integer Model 1	0.405	40	0.967	40	1.427	40
Rieck et al. (2012)	Mixed-Integer M1	0.029	40	0.153	40	0.199	40
	Mixed-Integer M2	0.029	40	0.108	40	0.206	40
	Tree-Based B&B	0.028	40	0.033	40	0.049	40
OVERLOAD							
This Study	B&B with NGD1	0.010	40	0.018	40	0.027	40
	B&B with NGD2	0.014	40	0.019	40	0.027	40
	Mixed-Integer Model 3	0.254	40	0.335	40	0.397	40
Rieck et al. (2012)	Mixed-Integer M3	0.031	40	0.140	40	0.298	40
	Mixed-Integer M4	0.014	40	0.027	40	0.051	40
	Tree-Based B&B	0.022	40	0.023	40	0.038	40
RID-MRD							
This Study	B&B with NGD3	0.013	40	0.171	40	0.064	40
	Mixed-Integer Model 4&5	61.609	40	105.809	36	429.105	31

Table 7.10 Computation results for *rlp_15* test sets

		rlp_15_1		rlp_15_3		rlp_15_5	
	Method	Time (s)	✓	Time (s)	✓	Time (s)	✓
SSQR							
This Study	B&B with NGD1	0.032	40	0.048	40	0.072	40
	B&B with NGD2	0.035	40	0.043	40	0.061	40
	Mixed-Integer Model 1	1.324	40	2.912	40	4.517	40
Rieck et al. (2012)	Mixed-Integer M1	0.154	40	0.291	40	0.513	40
	Mixed-Integer M2	0.154	40	0.356	40	1.194	40
	Tree-Based B&B	0.146	40	25.722	40	29.397	40
OVERLOAD							
This Study	B&B with NGD1	0.027	40	0.048	40	0.077	40
	B&B with NGD2	0.030	40	0.052	40	0.070	40
	Mixed-Integer Model 3	0.918	40	0.843	40	1.161	40
Rieck et al. (2012)	Mixed-Integer M3	0.309	40	0.975	40	14.390	40
	Mixed-Integer M4	0.096	40	0.071	40	0.112	40
	Tree-Based B&B	9.733	40	20.043	40	23.640	40
RID-MRD							
This Study	B&B with NGD3	0.156	40	0.228	40	1.461	40

When the computation results are compared, it is seen that the developed branch and bound algorithm has performed the best except for the test set of *rlp_30* for the results of SSQR and OVERLOAD metrics. Considering the tree-based branch and bound procedure stated at Rieck et al. (2012), the newly developed branch and bound procedure far more outperforms that algorithm. Also, the proposed branch and bound algorithm has revealed best performance for the test sets *rlp_10*, *rlp_15* and *rlp_20* based on all objective functions. For problem instances of 30 activities, integer linear modeling based procedures has proven better performance. The main advantage of branch and bound procedures is their flexibility to apply various techniques and to employ various objective functions. In integer linear modeling approach the type of the objective function is highly an effecting parameter for the performance. Computation time results of the branch and bound algorithm based on the RID-MRD function is not much different than other functions. Whereas, computation time results of the mixed-integer linear model developed for RID-MRD metric (results of both Table 7.4 and Table 7.9) are much worse than that of the branch and bound procedure. As stated in section 3.4, RID-MRD function generates bell-shaped resource utilization profiles where resource idle days with peak demand are minimized at the same time. For many construction projects, such utilization profiles may be more preferable. Both node gender decision methods NGD1 and NGD2 have performed similar, yet NGD2 has offered better performance in general. It has been preferred for most of the computational experimentation.

Table 7.11 Computation results for *rlp_20* test sets

		rlp_20_1		rlp_20_3		rlp_20_5	
	Method	Time (s)	✓	Time (s)	✓	Time (s)	✓
SSQR							
This Study	B&B with NGD1	0.190	40	1.259	40	1.445	40
	B&B with NGD2	0.282	40	0.468	40	0.968	40
	Mixed-Integer Model 1	6.714	40	73.061	40	90.689	40
Rieck et al. (2012)	Mixed-Integer M1	0.336	40	2.410	40	2.410	40
	Mixed-Integer M2	0.691	40	10.994	40	41.639	40
	Tree-Based B&B	1792.086	40	1540.256	40	3392.972	40
OVERLOAD							
This Study	B&B with NGD1	0.102	40	0.826	40	0.847	40
	B&B with NGD2	0.166	40	0.568	40	1.506	40
	Mixed-Integer Model 3	1.516	40	9.648	40	6.212	40
Rieck et al. (2012)	Mixed-Integer M3	1.236	40	16.086	40	48.214	40
	Mixed-Integer M4	0.191	40	1.172	40	1.523	40
	Tree-Based B&B	1566.697	40	2283.954	40	2751.409	40
RID-MRD							
This Study	B&B with NGD3	34.839	40	21.919	40	234.198	40

Table 7.12 Computation results for *rlp_30* test sets

		rlp_30_1		rlp_30_3		rlp_30_5	
	Method	Time (s)	✓	Time (s)	✓	Time (s)	✓
SSQR							
This Study	B&B with NGD1	308.662	37	183.297	39	338.914	33
	B&B with NGD2	308.919	39	54.427	40	401.747	36
	Mixed-Integer Model 1	258.116	39	285.705	39	347.955	35
Rieck et al. (2012)	Mixed-Integer M1	9.983	40	18.389	40	61.198	40
OVERLOAD							
This Study	B&B with NGD1	58.128	40	149.008	39	308.185	36
	B&B with NGD2	182.689	40	144.958	40	415.695	36
	Mixed-Integer Model 3	227.479	40	43.281	40	123.696	40
Rieck et al. (2012)	Mixed-Integer M4	12.529	40	14.155	40	101.538	40
RID-MRD							
This Study	B&B with NGD3	786.324	32	571.606	35	488.751	25

CHAPTER 8

CONCLUSION

There are few studies focusing on resource leveling problem. Most of these studies offer heuristic and metaheuristic methods to solve RLP. Despite heuristic and metaheuristic methods generate acceptable results for the RLP, it cannot be known if the results they generate are near optimal enough or not. In order to test the performance of heuristic and metaheuristic methods, exact solution results are required. Literature studies focusing on exact solution of resource leveling problem is even less than the heuristic and metaheuristic based studies. This thesis study aimed to improve existing exact solution methods and to develop an algorithm capable of solving larger problem instances to optimality. A branch and bound based procedure and a mixed-integer linear model have been developed.

The proposed branch and bound algorithm has been integrated with an adapted branch and bound heuristic. The adapted branch and bound heuristic utilizes very similar procedure of the main branch and bound algorithm to generate a good upper bound value at the beginning of the solution. This has improved the storage capacity consumption. The adapted branch and bound algorithm itself includes many accelerating techniques and offers different areas to study branch and bound methods for heuristic procedures. Another uniqueness of this thesis study is that a new lower bound calculation strategy that considers individual unscheduled activity resource effect has been developed to be used for sum of squares and overload resources metrics. The developed lower bound calculation strategy has been used together with the traditional lower bound calculation strategy. With the dual use of both lower bound calculations, it has been shown that the branch and bound algorithm can perform well for many problem sets of large size instances. Moreover, many computation improvement methods are developed for the proposed algorithm. By the implemented node gender decision systems, it has been observed that the selected unscheduled activity to be used for branching (node gender) highly affects the early marginal increase in lower bound values of newly generated nodes. Early higher marginal increase of lower bound value in newly generated nodes increases the possibility to reach more non-promising nodes at early stages thereby enables more pruning of the solution tree. A node gender decision system that considers maximum average resource effects of individual unscheduled activities also improves the computation time. A predefined activity priority list that best fits the specified problem instance can be utilized to determine the node gender activity. Candidate mother nodes are arranged from the best to the worst in a queue by an access of direct indexing method. Branching is performed from the best node by pulling out the first node of the

queue at each time. By the help of the queue, node extraction and placement operations have been so simplified that required operational time has been improved significantly.

Linear integer modeling has also been utilized in this thesis study. Resource leveling problem has been modeled with mixed-integer linear modeling based on the objective functions which are minimization of sum of squares, minimization of absolute deviations and minimization of resource idle days with maximum daily resource demand. Mixed-integer linear modeling has been also used in a very recent study by Rieck et al. (2012). In Rieck et al. (2012), the developed model could have solved RLP with the objective of minimization of sum of squares and overload resources.

Extensive computational experiments have been performed to test the proposed branch and bound algorithms. Results have indicated that the proposed branch and bound algorithm outperformed previous state-of-art branch and bound based methods significantly. The proposed branch and bound algorithm has outperformed the mixed-integer linear model developed by Rieck et al. (2012) for problems up to 20 activities. For larger size problems, with sum of squares and overload objective functions, the proposed branch and bound procedure still needs further improvements. However, as the performance of linear integer modeling based algorithms highly depend on the objective function, the proposed branch and bound algorithm is significantly better than the mixed-integer linear models for the RID-MRD objective function. The results for the RID-MRD experimentation have showed that the developed mixed-integer linear model in the concept of this thesis study could solve problems up to 10 activities in much longer durations compared to that of the branch and bound algorithm. The proposed branch and bound procedure could have solved problems up to 30 activities using RID-MRD objective function. As RID-MRD objective function profile is of practical significance for the construction projects, the branch and bound algorithm still offers an important base for the optimization of resource leveling problem. Moreover, since branch and bound procedure is flexible to be used with several applied techniques, it can be further studied to increase its capability to solve larger size actual construction projects to obtain leveled resource utilization curves.

As a future study, new lower bound calculation methods can be developed to improve the performance of branch and bound algorithms. An efficient lower bound calculation strategy significantly improves the solving capability of branch and bound algorithm. Besides, with the advance of the developing technology on multi-core processor chips, parallel computing techniques can be implemented to carry out multi processes on a single multi-core processing unit. By programming the algorithm suitable to parallel processing, computation times can be shortened to a considerable extent.

REFERENCES

Agin, N. (1966). "Optimum Seeking with Branch and Bpound", Management Science, Vol. 13, No. 4, pp. B-176-B-185.

Ahuja, H. (1976). Construction performance control by networks. New York: John Wiley and Sons.

Ballestin, F., Schwindt, C., and Zimmermann, J. (2007). "Resource Leveling in Make-to-Order Production: Modelling and Heuristic Solution Method", International Journal of Operational Research, Vol. 4, No. 1, 50-62.

Bandelloni, M., Tucci, M., and Rinaldi, R. (1994). "Optimal resource leveling using non-serial dynamic programming", European Journal of Operational Research 78, pp. 162-177.

Bettemir Ö. H. (2009). "Optimization of time-cost-resource trade-off problems in project scheduling using meta-heuristicalgorithms." PhD Thesis, Middle East Technical University, Ankara, Turkey.

Brucker, P., Knust, S., Schoo, A., and Thiele, O. (1998). "A Branch and Bound Algorithm for the Resource-Constrained Project Scheduling Problem", European Journal of Operational Research, Vol. 107, pp. 272-288.

Burgess, A. R., and Killebrew, J. B. (1962). "Variation in activity level on a cyclic arrow diagram.'' J. Industrial Engrg., 13(2), pp. 76–83.

Chan, W., Chua, K. H., and Kannan, G. (1996). "Construction resource Scheduling with Genetic Algorithms", Journal of Construction Engineering and Management, Vol. 122, No. 2, pp. 125-132.

Chen D., Lee C. Y., and Park C. H. (2005). "Hybrid genetic algorithm and simulated annealing (HGASA) in Global Function Optimization". Proceedings of the 17th IEEE International Conference in Tools with Artificial Intelligence (ICTAI'05), 1082- 3409/05.

Christodoulou, S. E., Ellinas, G., and Michaelidou-Kamenou, A. (2010). "Minimum Moment Method for Resource Leveling Using Entropy Maximization", Journal of Construction Engineering and Management, Vol. 136, No. 5, pp. 518-527.

De Reyck, B., and Herroelen, W. (1998). "A Branch and Bound Procedure for the Resource-Constrained Project Scheduling Problem with Generalized Precedence Relations", European Journal of Operational Research, Vol. 111, Issue 1, pp. 152-174.

Demeulemeester, E., and Herroelen, W. (1992). “A Branch-and-Bound Procedure for the Multiple Resource-Constrained Project Scheduling Problem”, Management Science, Vol. 38, No. 12, pp. 1803-1818.

Demeulemeester, E. (1995). “Minimizing Resource Availability Costs in Time-Limited Project Networks”, Management Science, Vol. 41, No. 10, pp. 1590-1598.

Demeulemeester, E., and Herroelen, W. (1997). “A Branch-and-Bound Procedure for the Generalized Resource-Constrained Project Scheduling Problem”, Operations Research, Vol. 45, No. 2, pp. 201-212.

Demeulemeester, E., and Herroelen, W. (2002). Project Scheduling: A Research Handbook, Kluwer Academic Publishers, Boston.

Doulabi, S. H. H., Seifi, A., and Shariat, S. Y. (2011). “Efficient Hybrid Genetic Algorithm for Resource Leveling via Activity Splitting”, Journal of Construction Engineering and Management, Vol. 137, No. 2.

Easa, S.M. (1989). “Resource leveling in construction by optimization”, Journal of Construction Engineering and Management, 115, pp. 302-316.

El-Rayes K., and Jun, D. H. (2009). “Optimizing resource leveling in construction projects”, Journal of Construction Engineering and Management, Vol. 135, No. 11, pp. 1172–1180.

Franck, B., Neumann, K., and Schwindt, C. (2001). “Truncated branchand- bound, schedule construction, and schedule improvement procedures for resource-constrained project scheduling”. OR Spektrum, 23, 297–324.

Gabow, H., and Myers, E. (1978). “Finding all spanning trees of directed and undirected graphs”. SIAM Journal on Computing, 7, pp. 280–287.

Galbreath, R. (1965). “Computer program for leveling resource usage”. Journal of the Construction Division, 91(1), pp. 107–124.

Gather T., Zimmermann J., and Bartels J. H. (2010), “Exact methods for the resource leveling problem”, Springer Science+Bussiness Media, LLC.

Geng J. Q, Weng L. P., and Liu S. H. (2010). “An improved ant colony optimization algorithm for nonlinear resource leveling problems”, Computers and Mathematics with Applications, doi:10.1016/j.camwa.2010.09.058 pp.1-6.

Guo, Y., Li, N., and Ye, T. (2009). “Multiple Resource Leveling in Multiple Projects Scheduling Problem Using Particle Swarm Optimization”, Fifth International Conference on Natural Computation, pp. 260-264.

Han M., Li P., and Sun J. (2006). “The algorithm for berth scheduling problem by the hybrid optimization strategy GASA”, Control, Automation, Robotics and Vision, 2006. ICARCV '06. 9th International Conference, 1-4.

Harris, R. (1978). “Precedence and arrow networks for construction”. New York: John Wiley and Sons.

Harris, R. B. (1990). “Packing Method for Resource Leveling (PACK)”, Journal of Construction Engineering and Management, Vol. 125, No. 3, pp. 167-175.

Hegazy, T. (1999). “Optimization of resource allocation and levelling using genetic algorithms.” Journal of Construction Engineering and Management, 125(3), pp. 167–175.

Hegazy, T., and Erşahin, T. (2001). “Simplified Spreadsheet Solutions: II: Overall Schedule Optimization”. Journal of Construction Engineering and Management, Vol. 127, No. 6, paper no 22158.

Hinze, J. W. (2004). Construction Planning and Scheduling, Pearson Prentice Hall, Upper Saddle River, New Jersey.

Hiyassat, M. A. S. (2000). “Modification of Minimum Moment Approach In Resource Leveling”, Journal of Construction Engineering and Management, Vol. 126, No. 4, pp. 278-284.

Hiyassat, M. A. S. (2001). “Appliying Modified Minimum Moment Method to Multiple Resource Leveling”, Journal of Construction Engineering and Management, Vol. 127, No. 3, pp. 192-198.

Hwang S. F., and He R.S. (2006). “Improving Real-Parameter Genetic Algorithm with Simulated Annealing for Engineering Problems”. Advances in Engineering Software 37: 406 – 418.

Icmeli, O., and Erenguc, S. S. (1996). “A Branch and Bound Procedure for the Resource Constrained Project Scheduling Problem with Discounted Cash Flows”, Management Science, Vol. 42, No. 10, pp. 1395-1408.

Jiang, G., and Shi, J. (2005). “Exact Algorithm for Solving Project Scheduling Problems under Multiple Resource Constraints”, Journal of Construction Engineering and Management, Vol. 131, No. 9, pp. 986-992.

Jun, D. H., and El-Rayes, K. (2011). “Multiobjective Optimization of Resource Leveling and Allocation during Construction Scheduling”, Journal of Construction Engineering and Management, Vol. 137, No. 12.

Karshenas S., and Haber D. (1990). “Economic Optimization of Construction Project Scheduling”, Construction Management and Economics”, Vol. 8, Issue 2, pp. 135-146.

Kartam, N., and Tongthong T. (1998). “An artificial neural network for resource leveling problems”, Artificial Intelligence for Engineering Design, Analysis and Manufacturing 12, 273–287.

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

Kolisch R., and Sprecher A. (1997). "PSPLIB – A project scheduling problem library", European Journal of Operational Research, Vol. (96), pp. 205 – 216.

Kolisch R., Schwindt A., and Sprecher A. (1999). "Benchmark Instances for Project Scheduling Problems", Operations Research and Management Science, Vol. 14, pp. 197-212.

Leu, S.S., and Yang, C.H. (1999). "GA-Based Multicriteria Optimal Model for Construction Scheduling", Journal of Construction Engineering and Management, Vol. 125, No. 6, pp. 420-427.

Leu, S.S., Yang, C.H., and Huang J. C. (2000). "Resource leveling in construction by genetic algorithm-based optimization and its decision support system application", Automation in Construction, 10, pp. 27–41.

Martinez, J., and Ioannou, P, (1993). ''Resource leveling based on the modified minimum moment heuristic.'' Proc., 5th Int. Conf., Computing in Civ. and Build. Engrg., ASCE, Reston, Va., pp. 287–294.

Mason, A. T., and Moodie, C. L. (1971). "A Branch and Bound Algorithm for Minimizing Cost in Project Scheduling", Management Science, Vol. 18, No. 4, pp. B-158-B-173.

Mattila, K. G., and Abraham, D. M. (1998). "Resource leveling of linear schedules using integer linear programming." Journal of Construction Engineering and Management, 124(3), pp. 232-244.

Mubarak, S. A. (2004). Construction Project Scheduling and Control, Pearson Prentice Hall, Upper Saddle River, New Jersey.

Mutlu, C. (2010). A Branch and Bound Algorithm for Resource Leveling Problem, Master Thesis, Middle East Technical University, Ankara, Türkiye.

Neumann, K., and Zimmermann J. (1999). "Resource Leveling for Projects with Schedule-Dependent Time Windows", European Journal of Operational Research, Vol. 117, pp. 591-605.

Neumann, K., and Zimmermann J. (2000). "Procedures for resource leveling and net present value problems in project scheduling with general temporal and resource constraints", European Journal of Operational Research, 127, pp. 425-443.

Neumann, K., Schwindt, C., and Zimmermann, J. (2003). Project Scheduling with Time Windows and Scarce Resources. Springer, Berlin.

Newitt, J. S. (2005). Construction Scheduling: Principles and Practices, Pearson Prentice Hall, Upper Saddle River, New Jersey.

Nübel, H. (2001). "The resource renting problem subject to temporal constraints". OR Spektrum, 23, 359–381.

Oral M., Oral E. L., Bozkurt S., and Erdiş E. (2003). "Yapım projelerinde genetik agoritma kullanarak kaynak seviyeleme", Çukurova Üniversitesi Mimarlık Fakültesi Dergisi, Cilt:18, Sayı: 2, pp.185-194.

Pang, N., Shi, Y., and You, Y. (2008). "Resource Leveling Optimization of Network Schedule Based on Particle Swarm Optimization with Constriction Factor", International Conference on Advanced Computer Theory and Engineering, pp. 652-656.

Patterson, J. H. (1984). "A Comparison of Exact Approaches for Solving the Multiple Constrained Resource, Project Scheduling Problem", Management Science, Vol. 30, No. 7, pp. 854-867.

Petrovic, R. (1969). On optimization of resource leveling in project plans. In H. J. Lombaers (Ed.), Project planning by network analysis: proceedings of the 2nd international congress (pp. 268–273). Amsterdam: North-Holland.

Ponz-Tienda, J. L., Yepes, V., Pellicer, E., and Moreno-Flores, J. (2013). "The Reosource Leveling Problem with multiple resources using an adaptive genetic algorithm", Automation in Construction, 29, pp. 161-172.

Qi J. X., Wang Q., and Guo X. Z. (2007). "Improved particle swarm optimization for resource leveling problem" Proceedings of the Sixth International Conference on Machine Learning and Cybernetics, Hong Kong, pp. 896-901.

Rieck, J., Zimmermann, J., and Gather, T. (2012). "Mixed-integer linear programming for resource leveling problems", Europian Journal of Operational Research, 221, pp. 27-37.

Roca, J., Pugnaghi, E., and Libert, G. (2008). "Solving an Extended Resource Leveling Problem with Multiobjective Evolutionary Algortihms", Proceedings of World Academy of Science, Engineering and Technology, Volume 36.

Savin D., Alkass S., and Fazio P., (1997). "A procedure for calculating the weight-matrix of a neural network for resource leveling" Advances in Engineering Software 28 28, pp. 271-283.

Senouci, A. B., and Eldin, N. N. (2004). "Use of Genetic Algorithms in Resource Scheduling of Construction Projects", Journal of Construction Engineering and Management, Vol. 130, No. 6, pp. 869-877.

Shah, K. A., Farid, F., and Baugh, J. W., Jr. (1993). "Optimal Resource Leveling Using Integer Linear Programming", Proceedings 4th International Conference on Computing in Civil and Building Engineering, pp. 501-508.

Son, J., and Skibniewski, M. J., (1999). "Multiheuristic approach for resource leveling problem in construction engineering: Hybrid approach." J. Constr. Eng. Manage., 125, pp. 23–31.

Son, J., and Mattila, K. G. (2004). "Binary resource leveling model: Activity splitting allowed." Journal of Construction Engineering and Management, 130(6), pp. 887-894.

Stevens, J. D. (1990). Techniques for Construction Network Scheduling, Mc Graw-Hill, New York.

Vanhoucke, M., Demeulemeester, E., and Herroelen, W. (2001). "On Maximizing the Net Present Value of a Project under Renewable Resource Constraints", Management Science, Vol. 47, No. 8, pp. 1113-1121.

Wang L., and Zheng D. (2001). "An effective hybrid optimization strategy for job-shop scheduling problems". Computers & Operations Research 28: 585 – 596.

Wiest, J., and Levy, F. (1977). "A management guide to PERT/CPM". Englewood Cliffs: Prentice Hall.

Woodworth, M. W., and Willie, C. J. (1975). "A Heuristic Algorithm for Resource Leveling in Multi-Project, Multi Resource Scheduling", Decision Sciences, Vol. 6, Issue 3, pp. 525-540.

Xiong, Y., and Kuang, Y. P. (2006). "Ant Colony Optimization Algorithm for Resource Leveling Problem of Construction Project", The CRIOCM 2006 International Symposium on "Advancement of Construction Management and Real Estate".

Younis, M.A., and Saad, B. (1996), "Optimal resource leveling of multi-resource projects", Computers and Industrial Engineering 31 pp. 1-4.

Zamani M. R. (2001). "A high-performance exact method for the resource-constrained project scheduling problem". Computers & Operations Research 28: 1387 – 1401.

Zheng, D. X. M., and Ng, S. T., Kumaraswamy, M. M. (2003). ASCE Construction Research Congress, pp. 1-8.

APPENDIX A

OPTIMAL RESULTS OF 21 PROBLEM SET (MUTLU, 2010)

Table A.1 Optimal results of the 21 problem set according to SSQR metric

SSQR (weighted sum of squares of daily resource demands)			All resources are weighted by 1. Optimal Result	
	Problem	Act.	Single Resource	Multi Resource
1	Demeulemeester (2002 Pg.416)	10	3522	9390
2	Easa (1989)	5	428	1096
3	El Rayes and Jun (2009)	20	3059	16863
4	Generated II (Mutlu 2010)	6	1238	3964
5	Generated III (Mutlu 2010)	14	1237	5049
6	Generated IV (Mutlu 2010)	18	1419	6303
7	Generated V (Mutlu 2010)	13	927	5540
8	Generated VI (Mutlu 2010)	14	1530	4442
9	Generated VII (Mutlu 2010)	16	3767	12802
10	Harris (1990)	11	821	3841
11	Hinze (2004 Pg.152)	15	509	3005
12	Leu (2000)	13	12246	33986
13	Mubarak (2004 Pg.61)	14	1817	10692
14	Mubarak (2004 Pg.67)	11	1553	6453
15	Mubarak (2004 Pg.217)	8	1636	5466
16	Newitt (2005 Pg.82)	16	1564	3720
17	Newitt (2005 Pg.121)	12	1043	5625
18	Son and Skibniewski (1999)	13	6225	18675
19	Son and Skibniewski (1999)	11	915	2801
20	Stevens (1990 Pg.97)	15	1525	8987
21	Stevens (1990 Pg.172)	19	2226	5608

Table A.2 Optimal results of the 21 problem set according to ABSDEV metric

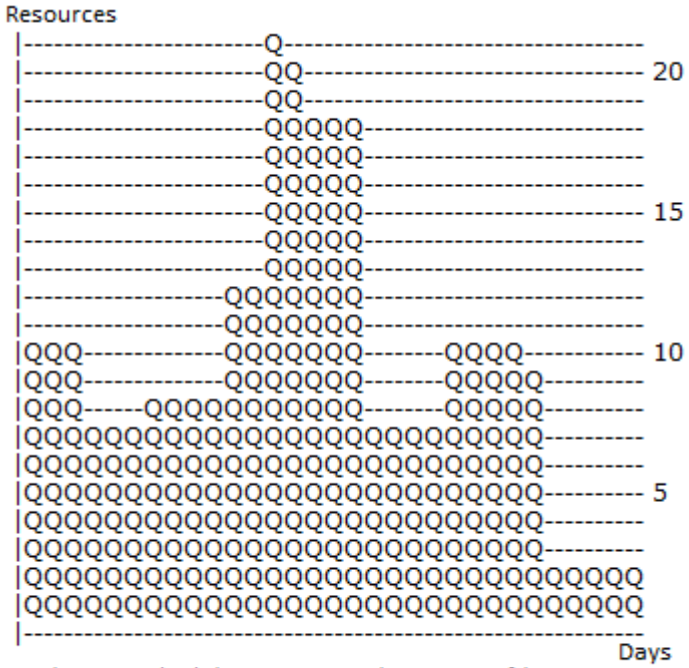
ABSDEV (weighted sum of absolute deviations from the average resource demand) (standard rounding)			All resources are weighted by 1. Optimal Result	
	Problem	Act.	Single Resource	Multi Resource
1	Demeulemeester (2002 Pg.416)	10	20	114
2	Easa (1989)	5	5	41
3	El Rayes and Jun (2009)	20	90	393
4	Generated II (Mutlu 2010)	6	18	98
5	Generated III (Mutlu 2010)	14	37	159
6	Generated IV (Mutlu 2010)	18	37	165
7	Generated V (Mutlu 2010)	13	27	198
8	Generated VI (Mutlu 2010)	14	45	238
9	Generated VII (Mutlu 2010)	16	75	366
10	Harris (1990)	11	25	141
11	Hinze (2004 Pg.152)	15	22	144
12	Leu (2000)	13	109	514
13	Mubarak (2004 Pg.61)	14	76	302
14	Mubarak (2004 Pg.67)	11	60	224
15	Mubarak (2004 Pg.217)	8	95	300
16	Newitt (2005 Pg.82)	16	48	206
17	Newitt (2005 Pg.121)	12	138	423
18	Son and Skibniewski (1999)	13	105	505
19	Son and Skibniewski (1999)	11	19	101
20	Stevens (1990 Pg.97)	15	67	337
21	Stevens (1990 Pg.172)	19	82	336

Table A.3 Optimal results of the 21 problem set according to RID-MRD metric

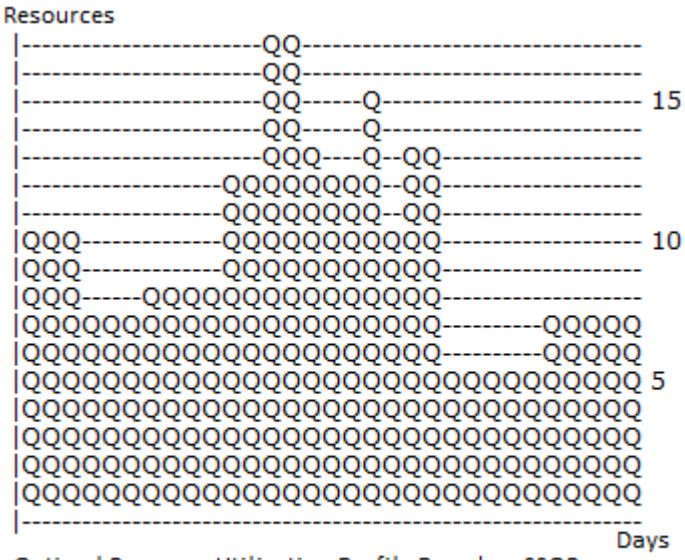
	RID-MRD (weighted sum of (1*resource idle days) + (1*maximum daily resource demand))		All resources are weighted by 1. Optimal Result	
	Problem	Act.	Single Resource	Multi Resource
1	Demeulemeester (2002 Pg.416)	10	19	69
2	Easa (1989)	5	10	43
3	El Rayes and Jun (2009)	20	17	200
4	Generated II (Mutlu 2010)	6	13	50
5	Generated III (Mutlu 2010)	14	45	108
6	Generated IV (Mutlu 2010)	18	25	110
7	Generated V (Mutlu 2010)	13	10	78
8	Generated VI (Mutlu 2010)	14	29	120
9	Generated VII (Mutlu 2010)	16	47	143
10	Harris (1990)	11	10	65
11	Hinze (2004 Pg.152)	15	13	84
12	Leu (2000)	13	31	132
13	Mubarak (2004 Pg.61)	14	11	75
14	Mubarak (2004 Pg.67)	11	11	80
15	Mubarak (2004 Pg.217)	8	12	58
16	Newitt (2005 Pg.82)	16	53	204
17	Newitt (2005 Pg.121)	12	23	80
18	Son and Skibniewski (1999)	13	22	43
19	Son and Skibniewski (1999)	11	9	91
20	Stevens (1990 Pg.97)	15	11	105
21	Stevens (1990 Pg.172)	19	14	171

APPENDIX B

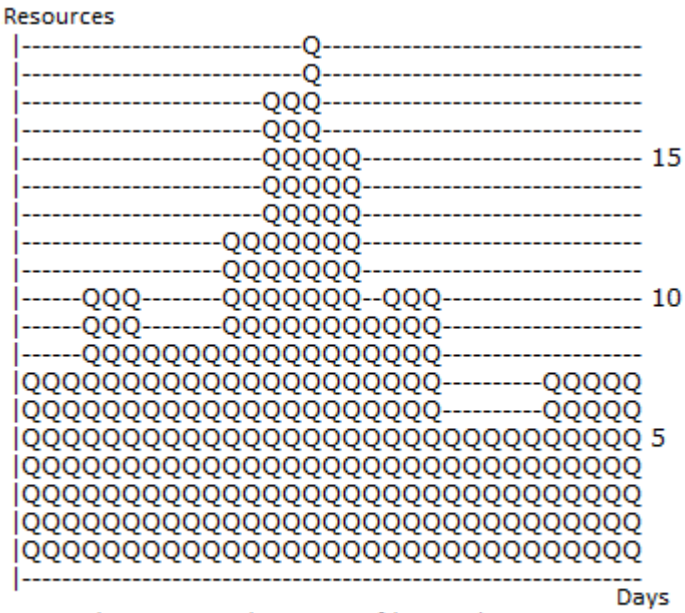
OPTIMAL RESOURCE UTILIZATION CURVES OF PROBLEM OF EL-RAYES AND JUN (2009)



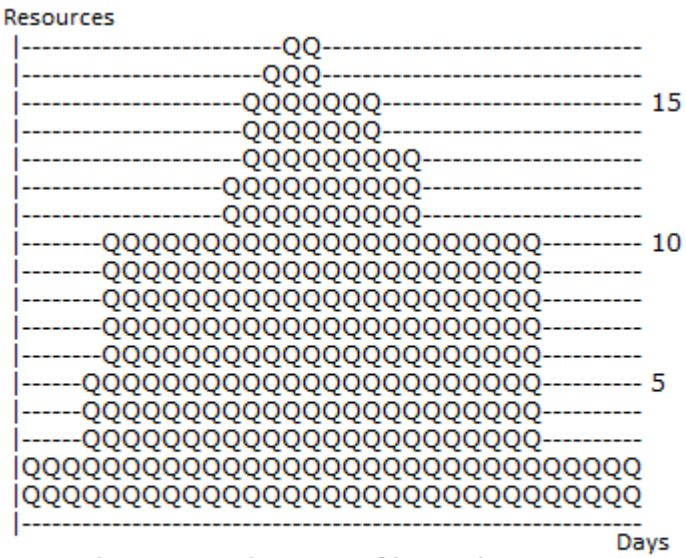
Early Start Schedule Resource Utilization Profile
(Values of SSQR: 3501, ABSDEV: 116, RID-MRD: 50)
El-Rayes and Jun (2009)



Optimal Resource Utilization Profile Based on SSQR
(Optimal Value: 3059)
El-Rayes and Jun (2009)



Optimal Resource Utilization Profile Based on ABSDEV
(Optimal Value: 90)
El-Rayes and Jun (2009)



Optimal Resource Utilization Profile Based on RID-MRD
(Optimal Value: 17)
El-Rayes and Jun (2009)