AN ABSTRACT OF THE DISSERTATION OF

Jirachai Buddhakulsomsiri for the degree of Doctor of Philosophy in Industrial Engineering presented on March 14, 2003.
Title: Multi-Mode Resource-Constrained Project Scheduling Problem with Resource Vacations and Task Splitting.

Abstract approved: Redacted for privacy
David S. Kim

The research presented in this dissertation addresses the Multi-Mode Resource-Constrained Project Scheduling Problem (MMRCPSP) in the presence of resource unavailability. This research is motivated by the scheduling of engineering design tasks in automotive product development to minimize the project completion time, but addresses a general scheduling situation that is applicable in many contexts.

The current body of MMRCPSP research typically assumes that, 1) individual resource units are available at all times when assigning tasks to resources and, 2) before assigning tasks to resources, there must be enough resource availability over time to complete the task without interruption. In many situations such as assigning engineering design tasks to designers, resources are not available over the entire project-planning horizon. In the case of engineering designers and other human resources, unavailability may be due to several reasons such as vacation, training, or being scheduled to do other tasks outside the project. In addition, when tasks are scheduled they are often split to accommodate unavailable resources and are not completed in one continuous time segment. The objectives of this research are to obtain insight into the types of project scheduling situations where task splitting may result in significant makespan improvements, and to develop a fast and effective scheduling heuristic for such situations.

A designed computational experiment was used to gain insight into when task splitting may provide significant makespan improvements. Problem instances were randomly generated using a modification of a standard problem generator, and optimally solved with

and without task splitting using a branch and bound algorithm. In total 3,880 problem instances were solved with and without task splitting. Statistical analysis of the experimental data reveals that high resource utilization is the most important factor affecting the improvements obtained by task splitting. The analysis also shows that splitting is more helpful when resource unavailability occurs in multiple periods of short duration versus fewer periods of long duration. Another conclusion from the analysis indicates that the project precedence structure and the number (not amount) of resources used by tasks do not significantly affect the improvements due to task splitting.

Using the insights from the computational testing, a new heuristic is developed that can be applied to large problems. The heuristic is an implementation of a simple priority rule-based heuristic with a new parameter used to control the number of task splits. It is desirable to obtain the majority of task splitting benefits with the smallest number of split tasks. Computational experiments are conducted to evaluate its performance against known optimal solutions for small sized problems. A deterministic version of the heuristic found optimal solutions for 33% of the problems and a stochastic version found optimal solutions for over 70%. The average percent increase in makespan compared to optimal was 7.58% for the deterministic heuristic and less than 2% for the stochastic versions demonstrating acceptable performance.

©Copyright by Jirachai Buddhakulsomsiri

March 14, 2003

All Rights Reserved

# Multi-Mode Resource-Constrained Project Scheduling Problem with Resource Vacations and Task Splitting

by

Jirachai Buddhakulsomsiri

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of

the requirements for the

degree of

Doctor of Philosophy

Presented March 14, 2003

Commencement June 2003

Doctor of Philosophy dissertation of Jirachai Buddhakulsomsiri presented on March 14, 2003.

APPROVED:

Redacted for privacy

Major Professor, representing Industrial Engineering

Redacted for privacy

Head of the Department of Industrial and Manufacturing Engineering

Redacted for privacy

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

Redacted for privacy

Jirachai Buddhakulsomsiri, Author

# ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor, Dr. David Kim for his support, guidance, and excellent advice during the past three years of this dissertation's work. I am also greatly indebted to the other two professors: Dr. David Butler, my advisor in Statistics, for his generous help and invaluable advice on this work; and Dr. Sabah Randhawa, my former advisor, for his constant support throughout the course of my study at Oregon State University.

I am very grateful to the other members of my dissertation committee: Dr. Daniel Schafer for his guidance on statistical aspect of this work; Dr. Kimberly Douglas and Dr. Jack Higginbotham for their useful comments and suggestions. I am also thankful to many people at the Statistics department: Dr. Fred Ramsey for being kind and always had answers to my questions whenever I knocked on his door; Dr. Jeff Arthur for his excellent teaching (and the lesson); Dr. David Birkes and Dr. Cliff Pereira for their suggestions on statistical analysis.

Many thanks to many friends who have helped me along the way in this long journey. Special thanks to Greg Jenkins for his statistical advice and SAS assistance; to all friends who have participated in the painful hand calculation experiment; to Wichai Chattinnawat and Parthana Parthanadee for sharing experiences and knowledge throughout the study.

Finally, I wish to thank my beloved family for the opportunity, unconditional love and support they have given me, knowing that doing so contributed greatly to my absence these last seven years. Nothing in a simple paragraph can express my deepest love and appreciation I have for them.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

Page

TABLE OF CONTENTS (Continued)

Page

# LIST OF FIGURES

# LIST OF TABLES

LIST OF TABLES (Continued)

LIST OF TABLES (Continued)

# LIST OF APPENDICES

Page

# LIST OF APPENDIX FIGURES

# LIST OF APPENDIX TABLES

# MULTI-MODE RESOURCE-CONSTRAINED PROJECT SCHEDULING PROBLEM WITH RESOURCE VACATIONS AND TASK SPLITTING

## 1. INTRODUCTION

### 1.1 OVERVIEW OF THE PROBLEM

This research addresses a generalization and properties of an important class of project scheduling problems known as Multi-mode Resource Constrained Project Scheduling problems. The generalization studied in this research is the preemption and resumption of tasks around unavailable resources (referred to as *task splitting*). Additionally, resource capacity variability caused by *resource vacations*, and the improvements attainable by task splitting, are studied in detail.

Multi-mode Resource Constrained Project Scheduling problems (MMRCPSP) occur in many real world applications. A very common characteristic of many real MMRCPSPs is that resources will not be available at all times, and that the periods of unavailability are known in advance. A motivating example will be presented in section 1.2, however real MMRCPSPs situations with periods of resource unavailability are easy to think of when human resources are involved. Human resources can become unavailable during a project due to vacations, special projects, training, or an unlimited number of other reasons, and these absences are usually known in advance. For non-human resources predictable absences often take the form of scheduled maintenance, overhauls, or use of a machine for a special task.

Most current tools and techniques for solving MMRCPSPs assume that a task, once started, cannot be interrupted. In other words, each task in a project can be scheduled only when the precedence constraints are satisfied and the required resources are available throughout the duration of the task. This assumption may lead to poor schedules, or schedules that can be significantly improved when tasks are split around unavailable resources. Moreover, splitting tasks in actual projects occurs in real problems (especially when human resources are involved) whether or not the "scheduling" tool permits it.

In general, a project consists of a set of tasks, governed by precedence relationships. The objective of interest is to complete the project in the minimal possible time allowed by the precedence relations. Project scheduling is less difficult when only precedence relations constrain the task scheduling. PERT (program evaluation and review technique) and CPM (critical path method) are techniques that are effective for these types of project scheduling problems. They provide the allowable time windows for scheduling the tasks in a project (computed in a polynomial time with respect to the number of tasks). Both techniques assume that resources of the appropriate types are sufficiently available so that resource capacities do not impact scheduling decisions. However, in most real situations, executing a task requires resources, and resources are available in limited amounts. The limitation of resources normally makes the project scheduling problems very difficult to solve optimally. When resources are constrained and each task can be executed in only one way (referred to as a "mode" and possibly requiring more than one resource type), the scheduling problem is known as the Resource Constrained Project Scheduling Problem (RCPSP). In many situations each task can be completed in one of several possible modes, with each mode possibly having a different duration, and potentially different resource requirements. This class of problems, which is a generalization of the RCPSP, is known as multi-mode resource-constrained project scheduling problems (MMRCPSP).

Project scheduling problems can be further distinguished by the types of resources considered. In the generalization of the MMRCPSP addressed in this research, only renewable resources are considered. Examples of renewable resources are people and machines. Resources such as these are normally available (renewed) for new tasks once a current task is completed. This is in contrast to non-renewable resources such as money, which is not available again after use.

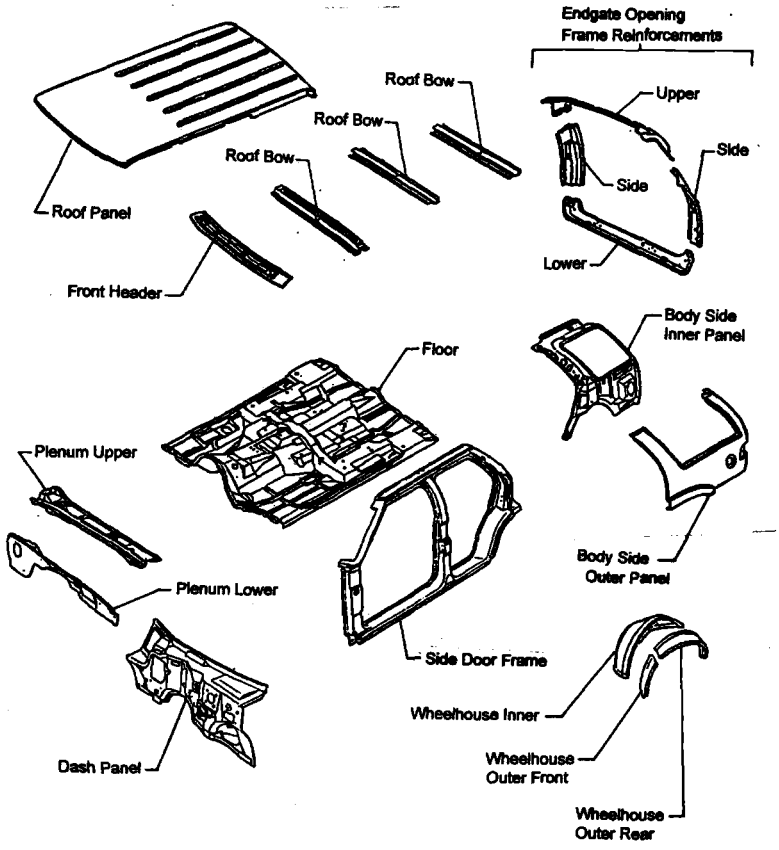
The motivation for studying this problem comes from many real world MMRCPSPs where there are potentially much better scheduling solutions available when task splitting is considered in a scheduling algorithm. Although task splitting reflects what occurs in reality, there has been relatively little research focused on solving this generalization of the MMRCPSP. Although a few groups of researchers have made contributions related to solving complex resource constrained project scheduling problems as shown in Bianco et al [1999], Demeulemeester and Herroelen [1996], and Valls et al. [1999], this work has addressed the single-mode case. Since this class of problems is known to be computational intractable, procedures that guarantee optimal solutions for large problems are not possible. Heuristic procedures for solving this particular problem on a large scale are also not yet available. Thus, there is indeed a need to study the problem in more depth and develop a heuristic procedure that is able to solve such scheduling problems effectively and efficiently.

## 1.2 MOTIVATING EXAMPLE

The example presented in this section is taken from the engineering design organization within General Motors (GM) Engineering. GM is the largest automobile company in the world and GM Engineering is the part of the company where new vehicle development is conducted. The Engineering Design function within GM Engineering is where the CAD for all vehicle components produced. This organization has a number of management

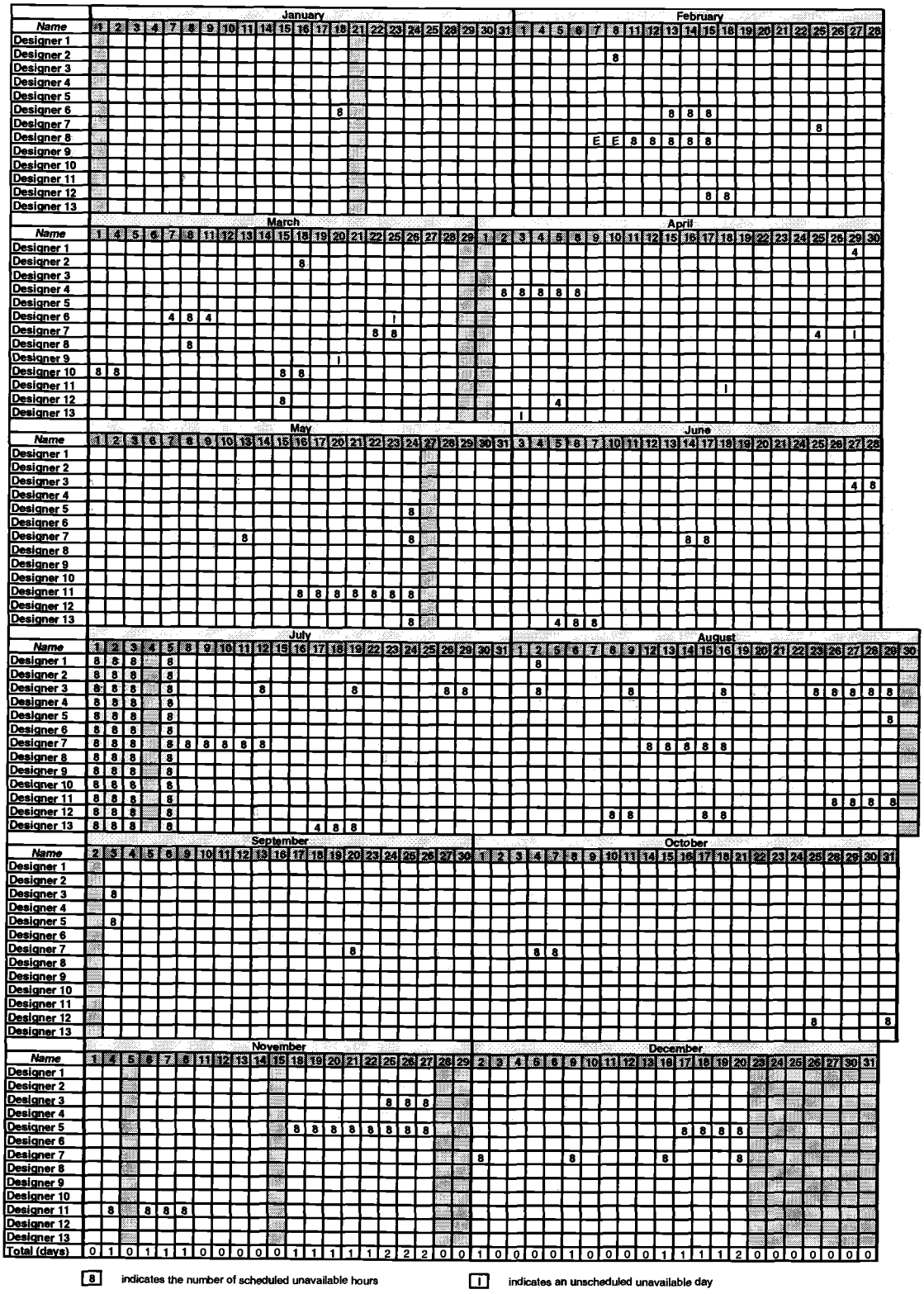
levels with the lowest level of management being the "design supervisor". Each design supervisors manages a group of design engineers responsible for doing CAD for vehicle components, and it is the responsibility of the design supervisor to assign CAD tasks to individual design engineers. Each vehicle component/subsystem consists of a number of parts that needs to be designed with known technological precedence relationships.

Figure 1.1 An example vehicle subsystem (vehicle body)




[1] Chih-Cheng Hsu, who is senior project engineer in the Operations Research Department at General Motors Engineering, provided this example.

Figure 1.2 One-year schedule of 13 designers in the group



An example[1] of a vehicle subsystem is shown in Figure 1.1. Once a design supervisor receives a CAD request for this subsystem, he or she defines a project consisting of tasks (i.e. designing each part making up the subsystem), precedence relations for the tasks, and the task due dates. Then, the supervisor needs to assign the tasks defined to designers in his/her design group. Designers in the group vary in skill level as well as the ability to perform tasks. From a scheduling perspective the different skill levels and abilities leads to different resource types. When considering the different resource types and the assigning of tasks to resources, the design supervisor normally has more than one way to assign each task, making this a multi-mode scheduling problem (see Hsu [2000] for more details).

Table 1.1 Summary of unavailable periods

| Name | Scheduled (day) | Unscheduled (days) | % Scheduled |
|---|---|---|---|
| Designer 1 | 5.5 | 0 | 100 |
| Designer 2 | 6 | 2 | 75 |
| Designer 3 | 21.5 | 0 | 100 |
| Designer 4 | 9 | 5 | 64.3 |
| Designer 5 | 19 | 0 | 100 |
| Designer 6 | 10 | 6 | 62.5 |
| Designer 7 | 28.5 | 3 | 90.5 |
| Designer 8 | 10 | 6 | 62.5 |
| Designer 9 | 4 | 0 | 100 |
| Designer 10 | 8 | 4 | 66.7 |
| Designer 11 | 19 | 0 | 100 |
| Designer 12 | 13.5 | 3.5 | 79.4 |
| Designer 13 | 10 | 0 | 100 |
| Total | 164 | 30 | 84.7 |

When assigning tasks to designers, the supervisor must deal with resources vacations (periodic resource unavailability). Designers go to training, take vacations, and also have

to do other tasks outside the project, and are therefore not available for the whole project planning period. Figure 1.2 is an example of the resource vacations over one calendar year for thirteen designers. For the schedule shown, the percent of potential project time that designers are unavailable is approximately 6.5 % on average. Most of the unavailable time for a resource is usually known in advance (among the unavailable time, 84.7% is known in advance). Table 1.1 summarizes the unavailable time for each designer within this year.

This real world example depicts the multi-mode resource-constrained project scheduling problem with resource vacations addressed in this research. It presents just one of many real world situations where task splitting should be taken into consideration when scheduling a resource-constrained project.

## 1.3 RESEARCH OBJECTIVES

This research has two main objectives.

1. Understanding general properties of resource vacations and when it is important to consider task splitting when scheduling tasks. In this research there are resource vacations, which result in variable resource capacities over time. The objective is to be able to determine under what project conditions task splitting is a beneficial component of a scheduling algorithm. This will be determined as a function of the parameters that describe the "vacation situation" and project situation. Examples of such parameters are the total percent of project time unavailable, the average length of a vacation, the location of vacations relative to the project start time, parameters describing the project, etc. Additionally, given that task splitting does improve the scheduling solution, how much improvement can be expected?

2. Develop a scheduling heuristic for the MMRCPSP that explicitly considers task splitting when assigning tasks to resources. This heuristic will take advantage of insights obtained from objective 1 as well as insights obtained from prior research, and an examination of problem structure. Computational experiments will be conducted to test the heuristic against exact results for smaller scheduling problems. Larger problems will also be tested but optimal solutions will not be available.

## 1.4 CONTRIBUTIONS OF THIS RESEARCH

This research has two major objectives. Both objectives contribute to the body of research in resource constrained project scheduling. Research objective 1 will provide fundamental insight into MMRCPSP by discovering what key parameters dictate when task splitting is important. Additionally, extensive computational experiments will be performed which will result in a library of scheduling problems and results. Objective 2 fills a clear gap in the body of scheduling heuristics available for resource constrained project scheduling problems and will provide insight into the behavior of MMRCPSPs in general.

## 1.5 DISSERTATION OUTLINE

The dissertation contains seven chapters. Chapter 2 is a review of the relevant literature. Chapter 3 gives a detailed description of the problem: problem statement and assumptions, mathematical model for the optimization problem, an example, complexity, and generalization of the problem.

Chapter 4 presents the project generator that has been widely accepted and used as a standard problem generator for this study, the exact algorithm used to solve small size problems to optimality, as well as two computational experiments on small size problems. These experiments are conducted to obtain some insight about the characteristics of the problem with resource vacations and task splitting. Chapter 5 includes heuristic methodology developed for solving medium to large-scale problems and two computational experiments, results, and analyses. These extended computational experiments are carried out to evaluate the performance of the developed heuristics. Chapter 6 features a real world problem that will be solved with the proposed heuristics. Finally, Chapter 7 concludes the study and suggests future research direction for the problem.

# 2. LITERATURE REVIEW

This chapter provides a summary of the relevant research and solution methodologies for the problem of interest. Section 2.1 reviews a classification scheme for the RCPSP, which can be used to organize RCPSP research. Section 2.2 presents an overview of RCPSP research. Section 2.3 reviews, in more detail, specific work that is closely related to the research in this thesis. Section 2.4 and Section 2.5 review exact algorithms for solving small-sized MMRCPSPs, and heuristic methods for solving large-scaled MMRCPSPs.

## 2.1 CLASSIFICATION OF RESOURCE-CONSTRAINED PROJECT SCHEDULING PROBLEMS

Since the class of RCPSPs encompasses a large number of specific problems, the potential for confusion and duplication of work exists. Different researchers have used different schemes of symbols and notation in order to denote their specific problem, which in some cases were actually the same. Without a standardized classification scheme using common notation, it is difficult to keep a clear view of the research area, directions, and progress. A few groups of researchers have attempted to provide such a classification scheme for the RCPSP (Brucker et al. [1999], Herroelen et al. [1999], and Ozdamar and Ulusoy [1995]).

A classification scheme for RCPSPs serves a variety of purposes. It facilitates the presentation and discussion on the subject, and allows researchers to identify viable research areas, which have remained unstudied or ignored in the field. It also helps identify the characteristics of the problem that one is working on. This section presents a classification scheme proposed by Herroelen et al. [1999]. Herroelen's classification scheme is adapted from a standard three-field notation, $\alpha|\beta|\gamma$, proposed by Graham et

al. in 1979 and Blazewicz et al. in 1983. It must be noted that not all parameters and possible values of each parameter are described here. Instead only the parameters that are relevant to this study are included. For a complete description of the classification scheme, readers are referred to Herroelen et al [1999].

### 2.1.1 Field $\alpha$

Field $\alpha$ refers to the resource characteristics of a RCPSP. Field $\alpha$ consists of at most three elements, $\alpha_1, \alpha_2$, and $\alpha_3$. Parameter $\alpha_1$ denotes the number of resource types, which may be ° (null), 1 resource type, or $m$ resource types. Parameter $\alpha_2$ denotes the specific resource types used. In RCPSPs, a common distinction is made between renewable resources, nonrenewable resources, doubly-constrained resources, and partially renewable resources.

Renewable resources, denoted by "1", is a resource type that is available for every period (e.g. hour, day, week, shift, etc.). Thus, use of renewable resources is constrained at every time period by the amount available. Nonrenewable resources, denoted by "$T$", are available for the entire project horizon, therefore, the use of nonrenewable resources is constrained over the total project horizon, and not at individual time periods. Doubly-constrained resources, denoted by "1$T$", are constrained every period and over the total project horizon. Partially renewable resources, denoted by "$\nu$", are resources where availability is associated with specific time periods, called period subsets. Partially renewable resources are renewed for every period subset, but they are nonrenewable within each period subset (Schirmer and Drexl [2001] and Bottcher et al. [1999]).

Examples of each resource type are: human resources or machines are often considered renewable resources, budget is typically considered a nonrenewable resource, and cash

consumed that is limited on a daily basis, but also constrained by a total project budget is a doubly-constrained resource. An example of a partially renewable resource is as follows: Suppose 5 machines (renewable) are available. One can define period subsets such as a week over the planning horizon. The availability of the five machines is renewed when each period subset starts, but within each period subset the use of the machines is nonrenewable.

Parameter $\alpha_3$ describes the resource availability characteristics of the RCPSP. $\alpha_3$ usually deals with renewable resources. It is defined as $\circ$, if renewable resources are available in constant amounts at every period and defined as $va$, if renewable resource availability may vary over time.

### 2.1.2 Field $\beta$

Field $\beta$ specifies the task characteristics of a RCPSP. It contains at most nine elements. $\beta_1$ is defined as $\circ$, when preemption is not allowed; or $pmtn$, when task preemption (task splitting) is allowed. This means that a task may be interrupted during processing and resumed at a later time. $\beta_2$ specifies the characteristics of the precedence constraints. If $\beta_2 = cpm$, then precedence constraints are strictly finish-start with zero time lag, as used in a PERT/CPM model. If $\beta_2 = gpr$, then precedence constraints are generalized precedence constraints of the type start-start, finish-start, start-finish, and finish-finish with both *minimum* and *maximum* time lags.

It is important to briefly describe the development of the generalized precedence constraints by De Reyck and Herroelen [1999]. They stated that the precedence constraints previously used in RCPSPs are finish-start precedence constraints with zero time lag and thus a special case of generalized precedence constraints. There are four

types of generalized precedence constraints (start-start, finish-start, start-finish, and finish-finish), with two types of time lag (maximum and minimum).

Under a start-start precedence constraint with minimum time lag a task can only start when its predecessor tasks have all started for a certain minimum time period. Under a start-finish precedence constraint with maximum time lag a task should be finished no later than a certain maximum number of time periods beyond the start of another activity (not necessarily its predecessors).

$\beta_3 =$ ° indicates that all tasks are ready at time zero, and $\beta_4 =$ ° indicates that all tasks have arbitrary integer (discrete) durations. $\beta_5 =$ °, indicates that there is no project deadline. $\beta_6 =$ ° indicates that all tasks require resources in a constant discrete amount.

$\beta_7$ is °, if each task has a single mode of execution. $\beta_7 = mu$, if tasks have multiple possible execution modes. $\beta_7 = id$, if tasks are subject to mode identity constraints. $\beta_7$ has taken on more possibilities as RCPSP research evolved. When RCPSPs were first introduced in 1960s, all tasks were assumed to be uniquely executed on one resource type (single-mode RCPSPs, Davis and Patterson [1975], Cooper [1976], and Christofides et al. [1987]). The concept of allowing multiple duration-resource options for each task was introduced in the 1970s, where task durations are no longer fixed but a function of the utilized resources (multiple-mode RCPSPs or MMRCPSPs, Slowinski [1981], and Talbot [1982]). More recently, Drexl et al. [1999] and Salewski et al. [1997] have generalized the multiple mode concept in what are called mode identity constraints. With mode identity constraints, the set of tasks is partitioned into disjoint subsets, where all tasks in a subset must be executed in the same mode.

### 2.1.3 Field $\gamma$

Field $\gamma$ indicates the performance measure of a project. $\gamma = reg$ denotes any measure based on project completion, and $\gamma = nonreg$, otherwise. For example $\gamma = reg$ includes RCPSPs with makespan (project completion time), flow time, or due date performance as measures. $\gamma = nonreg$ includes measures using resource leveling and financial measures as project performance measures. The minimization of the sum of squared deviation (or absolute deviation) of the resource requirements from the average, and the minimization of resource capacity to meet project due date are known as resource leveling problems (RLPs). Financial measures such as the maximization of net present value, the minimization project resource costs are known as resource investment problems (RIPs). $\gamma$ is not limited to single objective problems, but can also be augmented to include multiple-objective problems.

Examples of using this classification scheme to describe RCPSPs are presented next.

**Example 2.1** Consider a RCPSP with finish-start precedence relationships with zero time lag, renewable resources, project duration as a project measure, tasks with two execution modes having a fixed integer duration, constant resource requirements for each task executed in a mode, and no task splitting. This problem is denoted as $m,1|cpm,2|C_{\max}$.

**Example 2.2** The RCPSP in example 2.1 with task splitting allowed is denoted as $m,1|pmtn,cpm,2|C_{\max}$.

**Example 2.3** The RCPSP in example 2.2 with maximization of net present value of the project as project measure is denoted as $m,1|pmtn,cpm,2|npv$ (this problem is also known as a payment scheduling problem).

The RCPSP addressed in this research is denoted as $m,1,va \mid prmt,cpm,mu \mid C_{\max}$.

## 2.2 OVERVIEW OF RCPSP RESEARCH

Table 2.1 Documented RCPSP research.

Non-preemptive case

| Precedence relations | Modes | Resource type: Renewable resource only | Resource type: Renewable and non-renewable resources | Resource type: Partially renewable resoruce |
|---|---|---|---|---|
| CPM (finish-start with zero time lag) | Single | $m,1 \mid cpm \mid C_{\max}$ | $m,1T \mid cpm \mid C_{\max}$ | $m,v \mid cpm \mid C_{\max}$ |
| | Multiple | $m,1 \mid cpm,mu \mid C_{\max}$ | $m,1T \mid cpm,mu \mid C_{\max}$ | |
| | Mode identity | | $m,1T \mid cpm,id \mid C_{\max}$ | |
| Generalized precedence relations | Single | | | |
| | Multiple | | $m,1T \mid gpr,mu \mid C_{\max}$ | |
| | Mode identity | | | |

Preemptive case

| Precedence relations | Modes | Resource type: Renewable resource only | Resource type: Renewable and non-renewable resources | Resource type: Partially renewable resource |
|---|---|---|---|---|
| CPM (finish-start with zero time lag) | Single | $m,1 \mid pmtn,cpm \mid C_{\max}$ | | |
| | Multiple | $m,1,va \mid pmtn,cpm,mu \mid C_{\max}$ | | |
| | Mode identity | | | |
| Generalized precedence relations | Single | | | |
| | Multiple | | | |
| | Mode identity | | | |

Table 2.2 Examples of specific RCPSP research.

| Classification | Study |
|---|---|
| $m,1 \mid cpm \mid C_{max}$ | Alvarez-Valdes et al. [1989], Bell et al. [1991], Boctor [1990], Brucker et al. [1998], Christofides et al. [1987], Cooper [1976], Davis and Heidorn [1971], Davis and Patterson [1975], Demeulemeester and Herroelen [1992, 1995], Drexl [1991], Kolisch [1996a, 1996b], Kolisch and Hartmann [1999], Leon and Ramamoorthy [1995], Mingozzi et al. [1998], Oguz and Bala [1994], Patterson [1984], Pollack and Johnson [1995], Pritsker et al. [1969], Sampson and Weiss [1993], Schirmer and Riesenburg [1997], Shaffer et al. [1965], Simpson and Patterson [1996], Thomas and Salhi [1997], Ulusoy et al. [1989], Valls et al. [1992] |
| $m,1 \mid cpm,mu \mid C_{max}$ | Boctor [1993], Patterson et al. [1989], Patterson et al. [1990] |
| $m,1T \mid cpm \mid C_{max}$ | Kolisch et al. [1992, 1995], Kolisch and Sprecher [1996] |
| $m,1T \mid cpm,mu \mid C_{max}$ | Hartmann and Drexl [1997, 1998], Kolisch et al. [1992, 1995], Kolisch et al. [1999], Kolisch and Sprecher [1996], Slowinski [1981], Sprecher et al. [1997], Sprecher and Drexl [1998], Talbot [1982] |
| $m,1T \mid cpm,id \mid C_{max}$ | Drexl et al. [1999], Salewski et al. [1997] |
| $m,v \mid cpm \mid C_{max}$ | Bottcher et al. [1999], Drexl et al. [2000], Schirmer and Drexl [2001] |
| $m,1T \mid gpr,mu \mid C_{max}$ | De Reyck and Herroelen [1999] |
| $m,1 \mid pmtn,cpm \mid C_{max}$ | Bianco et al. [1999], Valls et al [1999], Demeulemeester and Herroelen [1996] |
| $m,1,va \mid pmtn,cpm,mu \mid C_{max}$ | This dissertation |

Table 2.1 and 2.2 lists the major RCPSP research conducted to date. Empty entries in the tables indicate that no documented research has been conducted in that area. For a complete list of research in each area, as of 1994, readers are referred to the survey study conducted by Ozdamar et al. [1995].

It can be seen from Table 2.2 that the majority of research has focused on the single mode RCPSP with only one resource type (renewable resource). There is relatively little research addressing RCPSPs with multiple modes and more than one type of resources. Even less published research considers the preemptive case. The next section will focus on research addressing the preemptive case.

## 2.3 RCPSP RESEARCH WITH TASK PREEMPTION

Bianco et al. [1999] considered the problem of scheduling tasks on dedicated resources, where each renewable resource can execute only one task at a time and each task can be preempted at integer points of time and resumed later without additional set up time. Their solution procedure is an exact algorithm using a graph-theoretical method based on new coloring techniques. Computational results are presented for lower bounds (computed by branch and bound algorithms) and upper bounds (computed by the Coloring Layered Network algorithm) of optimal solutions, where the performance of the proposed algorithm is evaluated based on those bounds. The assumption that one resource can be used to execute only one task at a time may be limiting, because in many situations resources consist of multiple interchangeable units, which can serve many task simultaneously.

Demeulemeester and Herroelen [1996] proposed a branch and bound procedure, based on the Demeulemeester and Herroelen [1992, 1995] algorithm (non-preemptive), for scheduling preemptive RCPSPs. The procedure employs a depth-first solution strategy in which nodes in the search tree represent precedence and resource feasible partial schedules. In order to apply the algorithm, each task in a project has to be broken into subtasks, each with unit duration. The number of subtasks is equal to the duration of the original task. A precedence based lower bound and five bounding rules are used to prune the search tree. Computational experiments on 220 test problems (110 from Patterson

[1984], and 110 from Simpson [1991]) is performed. It was found that scheduling with task splitting required an average of 33 times more computational time for the Patterson problems [1984] with an average makespan improvement of 0.78% A relative decrease in makespan of 2.88 was reported on the Simpson problems [1991] with a 12-fold increase in average computational time. Task splitting in both problems caused a considerable increase in the size of the search tree due to the growing number of subtasks. However, the small reduction in makespan reported led them to conclude that task splitting provides little benefit on scheduling except when resource availability is variable.

Valls et al. [1999] address a class of RCPSPs with stochastic task interruptions. A project consists of a set of deterministic tasks and a set of stochastic tasks. Some of the tasks may be interrupted by an uncertain amount of time. Each task that belongs to the stochastic set has an initial known duration and a known interruption time, however, the length of the interruption and the after-interruption duration are unknown. The proposed heuristic, called the scatter search method, schedules other tasks during the interruption of stochastic tasks to take advantage of idle resources. It consists of three decision stages, 1) at the beginning of the project; 2) after the completion of a task or the completion of the first part of a stochastic task; 3) at the end of a task interruption. At each decision point, the procedure selects a task based on a priority system that determines the next task to be scheduled. However, the heuristic does not focus on task splitting in general to improve a schedule. A computational experiment is performed and results are reported.

A comparison between RCPSP researches that considers task splitting is shown in Table 2.3. Demeulemeester and Herroelen [1996] reported a low percent improvement in makespan due to task splitting. These results may be due to the problems used in their computational experiments (they did their study at the same time as ProGen was being developed). They did not use a systematic system for generating examples problems tat spanned the problem space. The Valls et al. study [1999] has task splitting points

prespecified, while the length of interruptions are random. This setting clearly did not fully utilize the advantages of task splitting.

Table 2.3 Comparison of RCPSP research with task splitting.

| Research | Criteria | | |
|---|---|---|---|
| | Number of modes | Problem Instance source | Results |
| Bianco et al. | single | ProGen | 1. lower bound<br>2. upper bound<br>3. heuristic solutions for large-scaled problems |
| Demeulemeester and Herroelen | single | 1. Patterson [1984] (110 problems).<br>2. Simpson [1991] (110 problems) | 1. 110 optimal found with an average of 0.78% improvement in makespan.<br>2. 107 out of 110 optimal solutions found with an average of 2.88% improvement in makespan. |
| Valls et al. | single | ProGen | 1. heuristic solution for large-scaled problem |
| This research | multiple | ProGen | 1. optimal solutions for small problems.<br>2. heuristic solution for large problems. |

## 2.4 REVIEW OF EXACT ALGORITHMS FOR MMRCPSP

In general, solving a RCPSP with an exact algorithm implies that the problem will be solved to optimality. Optimal procedures include 1) dynamic programming (Kaplan [1988]); 2) 0-1 programming (Pritsker et al. [1969]); 3) graph representation approaches (Brucker et al. [1998]); 4) implicit enumeration with branch and bound (Davis and Heidorn [1971], Stinson et al. [1978], Christofides et al. [1987], Talbot [1982], Patterson et al. [1990], Demeulemeester and Herroelen [1992, 1995], Sprecher et al. [1997, 1998],

Hartmann and Drexl [1998], and Mingozzi et al. [1998]). Solving RCPSPs with exact algorithms is always limited in terms of the problem size due to the NP nature of the problem.

This brief discussion will focus on branch and bound algorithms only, because of their popularity and widely accepted efficiency and effectiveness. A variety of branch-and-bound procedures have been proposed for solving RCPSPs to optimality. Different ways of enumerating partial schedules have been developed. Various branch and bound algorithms have been developed by Stinson et al. [1978], Christofides et al. [1987], Demeulemeester and Herroelen [1992, 1995], Sprecher et al. [1995], and Mingozzi et al. [1998].

For multi-mode case all enumeration schemes with branch and bound procedures constructed to solve MMRCPSP use the concept of a partial schedule. The rest of this section presents a review of the three most competitive exact algorithms for MMRCPSPs: the precedence tree algorithm, the mode and delay alternative, and the mode and extension alternatives.

The precedence tree algorithm was first introduced by Patterson et al. [1989]. Sprecher and Drexl [1998] reconstructed the algorithm to be more efficient by including some bounding criteria. The mode and delay alternative procedure is a branch and bound approach proposed by Sprecher et al [1997]. It is an extension of the concept of delay alternatives used by Christofides et al. [1987] and Demeulemeester and Herroelen [1992] for the single mode RCPSP. The mode and extension alternative was developed by Hartmann and Drexl [1998]. It adopts the concept of mode alternatives developed by Sprecher et al. [1997] but uses a different way to extend partial schedules based on a method proposed by Stinson et al. [1978].
Hartmann and Drexl [1998] further simplified the precedence tree algorithm and provided a thorough comparison of the simplified precedence tree algorithms, the mode

and delay algorithms, and their mode and extension alternative. Solving a standard set of project instances generated by ProGen, it was found that the precedence tree algorithm by Sprecher and Drexl [1998] outperformed the other methods, in terms of the average and maximum computation times. It also was more efficient when solving "hard" problem instances and larger problems.

## 2.5 REVIEW OF HEURISTIC METHODS FOR MMRCPSP

This section reviews heuristic methods for MMRCPSPs. Kolisch and Hartmann [1999] surveyed and summarized most of the heuristics available in the literature. In this section two basic schedule generation schemes and some important heuristic methods are reviewed.

Most heuristic methods for RCPSPs are based on schedule generation schemes (SGSs). A SGS builds a feasible schedule by a stepwise extension of partial schedules, where a partial schedule is a schedule where a subset of all tasks has been scheduled. There are two main SGSs: serial SGSs and parallel SGSs. The serial SGS was first proposed by Kelley [1963]. It is a task incrementation procedure that consists of $J$ steps, where $J$ is the total number of tasks in a project. In each step, one task is selected and scheduled using a priority rule and a selection method. A priority rule is a mapping that assigns each eligible task a priority value. The selection method may be a deterministic or stochastic procedure that chooses a task to schedule based on the priority value of the eligible tasks. There has been large amount of research on priority rules for the RCPSP. Examples of priority rules that are widely used in the serial SGS are GRPW (greatest rank positional weight), Alvarez-Valdes and Tamarit [1989]; LFT (latest finish time), Davis and Patterson [1975]; LST (latest start time), Kolisch [1996b]; MSLK (minimum slack), Davis and Patterson [1975]; and MTS (most total successor), Alvarez-Valdes and Tamarit [1989]. The serial SGS terminates when all tasks are scheduled.

The parallel SGS was first proposed by Kelley [1963] and separately developed by Brooks in Bedworth and Bailey [1982]. The parallel SGS consists of at most $J$ steps. In each step, a schedule time $t_g$ is determined as the minimum finish time of all tasks that are already scheduled. Then, a set of tasks (which may be empty), are selected and scheduled using one priority rule and a selection method. Priority rules that are widely used in the parallel SGS include rules used in serial SGSs and the following rules that are only applicable in a parallel SGS: RSM (resource scheduling method), Shaffer et al. [1965]; IRSM (improved RSM), Kolisch [1996b]; WCS (worst case slack), Kolisch [1996b]; and ACS (average case slack), Kolisch [1996b]. The parallel SGS is terminated when all tasks are scheduled.

Heuristic approaches, developed based on SGSs, include: 1) priority rule based scheduling (single-pass and multi-pass); 2) truncated branch and bound procedures (Alvarez-Valdes and Tamarit [1989], and Pollack and Johnson [1995]); 3) disjunctive arc concepts (Shaffer et al. [1965], Alvarez-Valdes and Tamarit (1989), and Bell and Han [1991]); and 4) local search techniques and metaheuristic methods (Sampson and Weiss [1993], Leon and Balakrishnan [1995]).

Priority rule-based scheduling with serial and parallel SGSs are the two best-known and oldest heuristics for solving RCPSPs. Even though they are outperformed by more recent heuristics, they are still useful due to their simplicity and effectiveness. Additionally, the results from these heuristics are used as starting points for more advanced search algorithms (metaheuristics).

Most publications on priority rule-based scheduling methods focus on either developing new priority rules and improving existing rules, or developing new selection methods. Kolisch [1996a] provides a review of serial and parallel SGSs that includes all priority rules known to date. Kolisch [1996a] conducted an extensive computational study to investigate the relationship of single-pass scheduling (deterministic selection) and

sampling (stochastic selection) for both SGSs. The results show that: 1) serial and parallel SGS perform similarly, 2) the performance ranking of priority rule does not differ for single-pass scheduling and sampling, and 3) sampling improves the performance of single-pass scheduling significantly. Kolisch [1996b] proposed improved priority rules and conducted an in-depth comparison of their performance, and also ranked a variety of priority rules. Other research where priority rules are assessed are: Davis and Patterson [1975] compared three rules and reported their ranking as MSLK>LFT>RSM (">" indicates better performance); Boctor [1990] reported the same results as Davis and Patterson [1975]; Alvarez-Valdes and Tamarit [1989] evaluated another set of three rules and reported the ranking of GRPW>LFT>MTS.

The truncated branch and bound method was first developed by Alvarez-Valdes and Tamarit [1989]. More recently, Pollack and Johnson [1995] used a depth-first, "jumptracking" branch and bound search of the partial solution tree. The algorithm is basically a parallel scheduling heuristic, which performs branching according to a priority value (i.e. one branch has the task with highest priority value and the other branch has the task with the second highest priority value). Sprecher [1996] employs a depth-first search branch and bound as a heuristic by imposing a time limit on the algorithm. In order to obtain good solutions early in the search process, priority rules are applied to select the most promising task from the decision set for branching.

Disjunctive arc based approaches are performed by extending the precedence relationships by adding additional arcs such that "minimal forbidden sets" (i.e. sets of independent tasks which cannot be scheduled simultaneously due to resource constraints) are eliminated, so that the earliest finish schedule is precedence and resource feasible. Shaffer et al. [1965] first proposed this method to be used within the forbidden sets for which all tasks in the earliest finish can be processed at the same time. The disjunctive arc that produces the smallest increase in the earliest finish time of the unique sink (in the forbidden set) is introduced. The algorithm terminates once a precedence and resource

feasible earliest finish schedule is found. Alvarez-Valdes and Tamarit [1989] later proposed four different methods of destroying the minimal forbidden sets. They found that the best results were obtained by choosing the lowest cardinality forbidden set and destroyed it by adding the disjunctive arc that produces the minimum earliest finish time of a unique dummy sink (the last task of the project). Bell and Han [1991] proposed a two-phase algorithm to further improve this method by first applying Shaffer et al. [1965] approach to obtained a feasible solution (phase one) and then reapplying the method.

There have been a number of metaheuristic procedures developed for RCPSPs in the literature. Kolisch and Hartmann [1999] provide a discussion of such research. These metaheuristic methods are based on applying one of the three metaheuristics: simulated annealing (SA), TABU search (TS), and genetic algorithms (GA), which use initial solutions generated by other heuristics. Most published methods are applicable to single-mode RCPSPs. Recently, Kolisch and Hartmann [1999] performed a comparison of available heuristics on a set of large problem instances. Their results show that the genetic algorithm of Hartmann [2002] is the most effective method. In addition, Hartmann and Drexl [1997] and Hartmann [2001] developed a metaheuristic for solving multi-mode problems using GAs. Their algorithm is a powerful heuristic for solving MMRCPSP.

# 3. PROBLEM DESCRIPTION

This chapter presents a description of the research problem and its characteristics. In section 3.1 the problem description and assumptions are given. Section 3.2 presents a formal mathematical model of the MMRCPSP. Section 3.3 gives a small example of the MMRCPSP. In section 3.4 the problem is classified and the computational complexity of the problem is discussed.

## 3.1 PROBLEM STATEMENT AND ASSUMPTIONS

### 3.1.1 Problem Statement

A project to be scheduled consists of a set of $J$ tasks. All tasks in the project will require one or more of the $R$ types of renewable resources. The project must be completed by its deadline. The order in which some tasks are performed is specified by the technological precedence relationship. Otherwise, they can be done in any order. Each task $j$ can be performed in one of $M_j$ executable modes. Each executable mode may have different resource requirements and resource demands. Thus, the same task performed in different modes may take different durations to complete. There are $R$ resource types, each with $K_r$ resource units. Resource units within each type of resource may not be available during some periods of the project-planning horizon. This unavailability is known in advance and specified in a vacation (unavailability) schedule. Tasks, once started, may be interrupted due to the schedule of the resource unit assigned to the tasks. The objective is to complete the project as soon as possible.

### 3.1.2 Problem Assumptions

The MMRCPSP described also adheres to the following assumptions.

(A) A project consists of $J$ tasks, represented using an activity-on-node format (AON), where nodes represent the tasks, and arcs denote the precedence relationships. Two dummy tasks are introduced. Dummy task 1 represents the start task of the project and dummy task $J$ represents the end task of the project.
(B) Precedence relationships are *finish-start* with a time lag of zero, meaning a task can be started if and only if all of its predecessors have completed.
(C) Each task $j$ can be performed in one of $M_j$ possible modes, where each task-mode combination has a fixed duration. Each mode requires a constant amount of one or more of the $R$ types of renewable resources for the entire task duration.
(D) Every resource unit within each type of resource has a known vacation schedule.
(E) When tasks are split, they are resumed without additional duration.
(F) Mode switching is not allowed when tasks are split.
(G) The objective is to complete the project as soon as possible.

## 3.2 MATHEMATICAL MODEL OF THE MMRCPSP

A mathematical programming formulation of the MMRCPSP described is presented next.

Minimize $F_J$ (3.1)

Subject to:

$$\sum_{m=1}^{M_j} y_{j,m} = 1 \qquad \forall j \in \{2,..,J-1\} \qquad (3.2)$$

$$\sum_{t=1}^{T} x_{j,m,t} = d_{j,m} \times y_{j,m} \qquad \forall j \in \{2,..,J-1\}, \forall m \in \{1,..,M_j\} \qquad (3.3)$$

$$\max_{t=1}^{T} \left( x_{j,m,t} \times t \right) = F_j \qquad \forall j \in \{2,..,J\} \qquad (3.4)$$

$$\min_{\substack{t=1 \\ \{x_{j,m,t}>0\}}}^{T} \left( x_{j,m,t} \times t \right) = S_j \qquad \forall j \in \{2,..,J-1\}, \forall m \in \{1,..,M_j\} \qquad (3.5)$$

$$F_i < S_j \qquad \forall (i,j) \in P_j \qquad (3.6)$$

$$\sum_{j=2}^{J-1} \sum_{m=1}^{M_j} \left( k_{j,m,r} \times x_{j,m,t} \right) \le K_{r,t} \qquad \forall r \in \{1,..,R\}, \forall t \in \{1,..,T\} \qquad (3.7)$$

$$y_{j,m} \in \{0,1\} \qquad \forall j \in \{2,..,J-1\}, \forall m \in \{1,..,M_j\} \qquad (3.8)$$

$$x_{j,m,t} \in \{0,1\} \qquad \forall j \in \{2,..,J-1\}, \forall t \in \{1,..,T\}, \forall m \in \{1,..,M_j\} \qquad (3.9)$$

$$S_j \in \{0,1,2,...\} \qquad \forall j \in \{2,..,J-1\} \qquad (3.10)$$

$$F_j \in \{0,1,2,...\} \qquad \forall j \in \{2,..,J-1\} \qquad (3.11)$$

The index sets in the model are:

$i, j$ the task index, $j \in \{2,..,J\}$

$m$ the mode index, $m \in \{1,..,M_j\}$

$t$ the time period index, $t \in \{1,..,T\}$

$r$ the resource index, $r \in \{1,..,R\}$

The parameters are:

$J$ the total number of tasks in the project

$M_j$ the number of modes for task $j$

$d_{j,m}$ duration of task $j$ if it is executed in mode $m$

$T$ the upper bound of the project completion time, $T = \sum_{j=2}^{J-1} \max_{m=1}^{M_j} \{d_{j,m}\}$

$R$ the total number of renewable resources in the project

$P_j$ the set defining precedence relationship of task $j$; $P_j$ consists of pairs of $(i, j)$, where $i$ is a predecessor of task $j$.

$k_{j,m,r}$ units of resource $r$ required by task $j$ if it is executed in mode $m$

$K_{r,t}$ the capacity of renewable resource $r$ available for period $t$

And the decision variables are:

$y_{j,m}$ binary variable represents whether task $j$ is being executed in mode $m$.

$x_{j,m,t}$ binary variable represents whether task $j$, executed in mode $m$, is consuming any resource at time $t$ (i.e. whether task $j$ is "in-progress" at time $t$).

$S_j$ general integer variable represents the start time of task $j$.

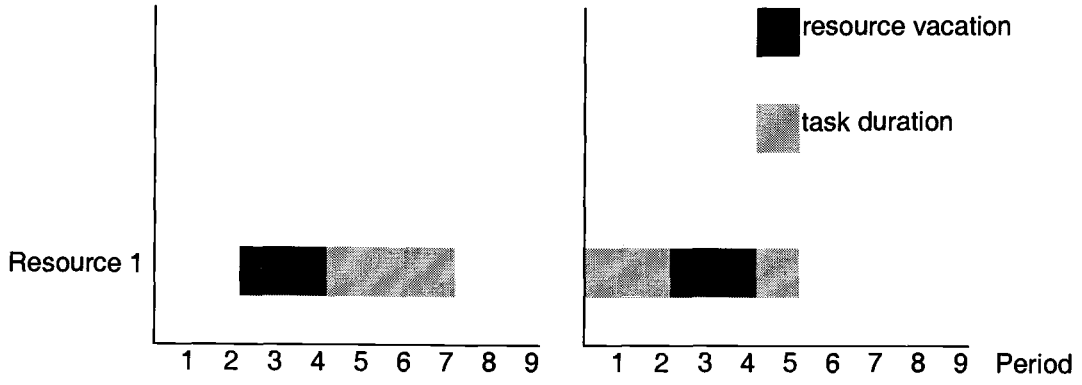
$F_j$ general integer variable represents the finish time of task $j$.

In the formulation, the objective function (3.1) minimizes the project makespan. Constraint (3.2) ensures that each task $j$ is executed in only one mode. Constraint (3.3) ensures that each task $j$ is executed only once, and that the total number of periods that task $j$ uses resources is equal to the duration of that task when executed in mode $m$. Constraints (3.4) and (3.5) represent the finish time and start time for each task $j$, respectively. Constraint (3.6) represents the finish-start precedence relationships. Constraint (3.7) forces the total units of resource utilized to be less than or equal to the available resource capacity for every period.

The potential impact of the task splitting on a schedule is shown in Figure 3.1. The schedule on the left in Figure 3.1 shows that resource 1 is not available during periods 3 and 4. If task splitting is not allowed, task $j$, which takes 3 periods to complete, cannot be scheduled to start at period 1 since there is a time window of only 2 periods with sufficient resources. Task $j$ will be completed at the end of period 7. The schedule on

the right shows that splitting task $j$ allows it to be started at period 1 and completed at the end of period 5.

Figure 3.1 The effect of task splitting on a schedule




Intuitively, it can be seen that task splitting as defined for this study increases the elapsed time to finish a task (finish time minus start time). This increase in elapsed time however, may result in an earlier task completion time.

## 3.3 AN EXAMPLE MMRCPSP

In this section an example project is presented. An optimal schedule for this example is shown when there are no resource vacations. Another optimal schedule is shown when there are resource vacations and tasks are not split, demonstrating the impact of resource unavailability. An optimal schedule with task splitting permitted is then shown to demonstrate the potential benefits of task splitting in the presence of resource vacations.

### 3.3.1 Example Problem

A project consists of 10 tasks with precedence relationships as shown in Figure 3.2. Task 1 and task 12 are dummy tasks with zero duration. The project deadline is 24, and there are two types of renewable resources used in this project. To simplify the example, each type of resource has a maximum of one unit available in any time period. There are two modes for available for executing each task. The task durations for each task-mode combination are shown in Table 3.1 (generated from a discrete uniform distribution between 1 and 7 time periods).

Figure 3.2 Precedence relationships for the example project

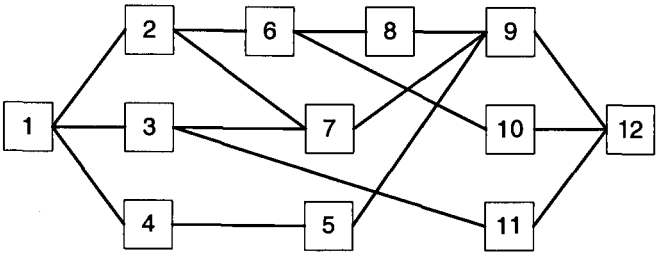



Table 3.1 Task-mode durations

| Mode | Task | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 1 | 2 | 4 | 1 | 2 | 4 | 2 | 2 | 7 | 2 | 7 |
| 2 | 3 | 4 | 1 | 2 | 2 | 5 | 6 | 5 | 4 | 2 |

The resource demands for each task-mode combination are shown in Table 3.2 (generated from a binary distribution with the restriction that each combination needs one type of resource).

Table 3.2 Task-mode resource demands

| Mode | Resource | Task | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| | 2 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 2 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| | 2 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |

### 3.3.2 Optimal Schedules Without and With Resource Vacations

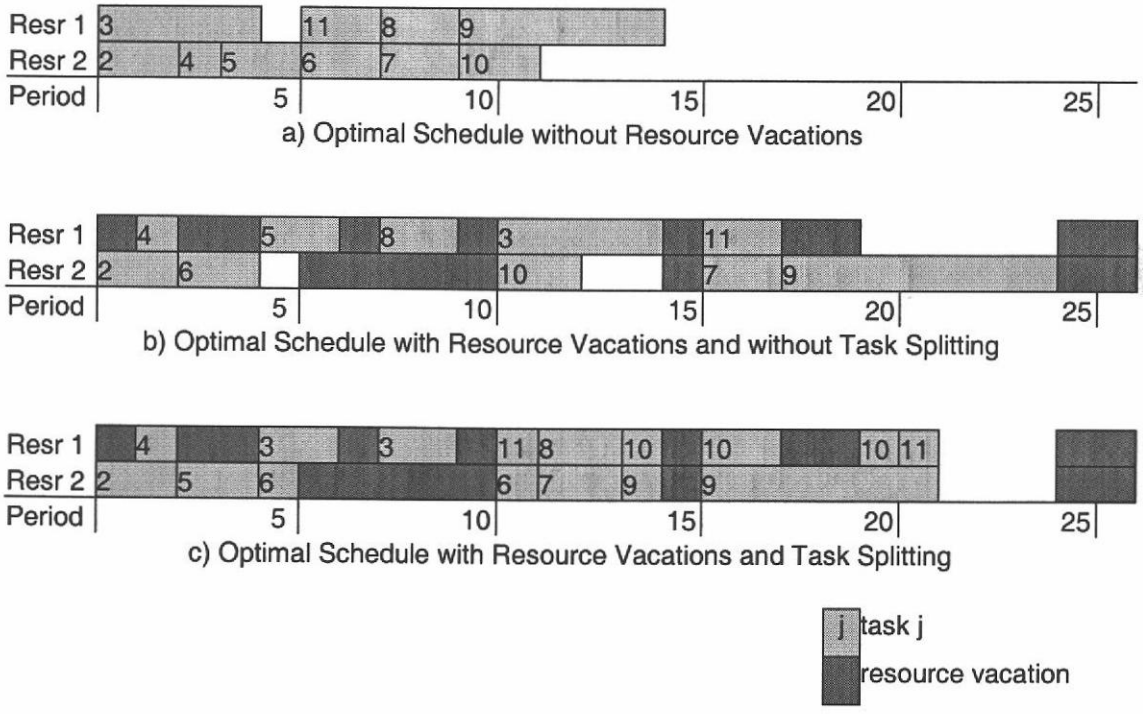
The example was solved optimally with an implicit enumeration method (presented in Chapter 4). The optimal schedule has a makespan of 14 time periods when each resource is available at all times over the project horizon. With a set of randomly generated resource vacations as shown in Figure 3.3 b) the optimal makespan increases to 24 time periods. The optimal solutions for both cases are shown in Figure 3.3 a) and b). The increase in the optimal solution of 10 is greater than the total resource unavailability imposed on both resources (only resource vacations that come before the project makespan are considered). This demonstrates the potential negative impact of resource vacations on the optimal schedule.

### 3.3.3 Scheduling Around Resource Vacations With Task Splitting

The results from the previous section show that presences of resource vacations have a large negative impact on the optimal schedule. This section illustrates the potential benefits of task splitting when scheduling in the presences of resource vacations. The previous example with resource vacations from the previous section is solved optimally with task splitting permitting, resulting in a makespan of 21. The optimal schedule with

task splitting is shown in Figure 3.3 c). The optimal makespans for all cases are shown in Table 3.3.

Figure 3.3 Optimal schedules for all 3 cases




In the presence of resource vacations in this example, the effect of task splitting is an optimal makespan reduction of 3, which is a 12.5% improvement. However, because this example is only one problem instance, it cannot be concluded that task splitting will always result in a reduced optimal makespan. In this research an investigation with well-design computational experiments will be conducted to explore what types of improvements are typical for a variety of different problems.

Table 3.3 Summary of results

| | without resource vacations | with resource vacations |
|---|---|---|
| without task splitting | 14 | 24 |
| with task splitting | 14 | 21 |

## 3.4 CHARACTERISTICS OF THE GENERALIZED PROBLEM

### 3.4.1 Complexity of the Problem

The MMRCPSP with resource vacations and task splitting is NP-hard with respect to computational complexity. This can be verified in the following way. It has been proved that finding a feasible solution for the RCPSP with a due date constraint is NP-Complete (Garey and Johnson [1979]). Since the RCPSP is a special case of the MMRCPSP, finding the feasible solution for the MMRCPSP with resource vacations and task splitting must also be NP-Complete. Thus, the search for an optimal schedule solution for this problem is NP-Complete.

### 3.4.2 Classification of the Problem

The MMRCPSP of interest can be classified as $m,1,va \mid prmt,cpm,mu \mid C_{\max}$, where $va$ denotes the availability of resources in a variable manner, and $prmt$ denotes the general splitting of tasks, i.e. tasks can be interrupted, and once interrupted, can be resumed at any later time. The introduction of task splitting into the MMRCPSP increases the number of feasible solutions. This is because at every time period, decisions on what tasks are to be scheduled for one time period must be made. This makes the number of

possible tasks that can be scheduled at each period much larger than when task splitting is not permitted.

### 3.4.3 Activity-On-Node Representation of the Generalized Problem

To represent the $m,1,va \mid prmt,cpm,mu \mid C_{\max}$ problem via an Activity-On-Node (AON) graph, the concept of subtask must be introduced (Demeulemeester et al. [1996]). A task is replaced by one or more subtasks, each with unit duration such that the number of subtasks is equal to the duration of the original task. Each subtask has the same resource requirements and task-mode options as the original task. Splitting tasks into subtasks increases the number of decision variables in the model, which makes the already NP-complete problem much larger.

Figure 3.4 represents an AON of a MMRCPSP. Each block represents a task, with its duration given above each block. Each arc denotes finish-start precedence relationships. By splitting each task into subtasks, the original AON is transformed into the one in Figure 3.5, where each subtask has unit duration.

Figure 3.4 Original AON

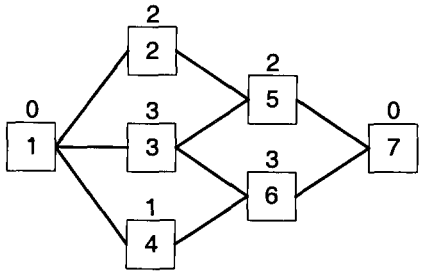
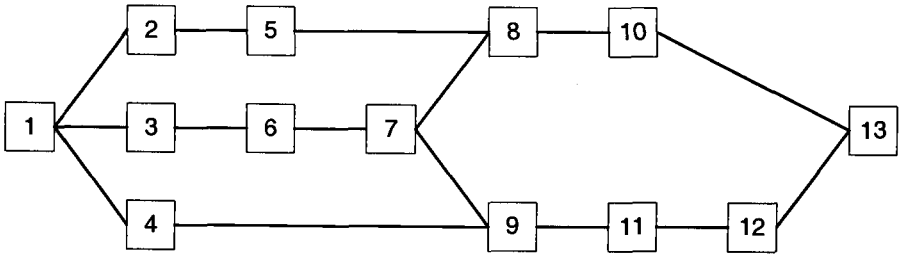


Figure 3.5 Transformed AON



# 4. COMPUTATIONAL INVESTIGATION OF TASK SPLITTING IN THE PRESENCE OF RESOURCE VACATIONS

To investigate the impact of task splitting in MMRCPSPs with resource vacations, a large computational experiment was conducted. In this chapter the various components required to execute such a study are presented, as well as the results of the experiments. This chapter starts with a description of the ProGen algorithm for generating project scheduling problems, and how ProGen is modified to include resource vacations. In Section 4.2, the exact algorithm (implicit enumeration method) that is used to solve small size problems to optimality, both without task splitting and with task splitting, is described. In section 4.3 and section 4.4, the two computational experiments conducted are described and their results presented. For each experiment, the objective, design, summary of results, and statistical analysis are presented. Insights learned from the experiments are summarized in Section 4.4.

## 4.1 MODIFIED PROJECT GENERATOR

This section describes a problem instance generator for RCPSPs. Project generator (ProGen), developed by Kolisch et al. [1992, 1995], is a standard project instance generator, which is based on a set of parameters that capture many aspects of a general RCPSP. With ProGen, the generation of project instances is systematic, assuring that a set of projects can be generated that represents the total "space" of projects. Since ProGen was introduced it has been used in a number of different published research studies. Several other versions of ProGen have been developed by researchers for their own problems, such as ProGen/πx for RCPSPs with partially renewable resources (Drexl et al. [2000]), ProGen/max for RCPSPs with generalized precedence constraints (De Reyck and Herroelen [1999]), etc. Standard problem sets for various RCPSPs, (RCPSP,

MRCPSP, RCPSP/max, MRCPSP/max, RIP/max, RLP/max) generated by various ProGen's are included in the project scheduling library, PSPLIB (Kolisch and Sprecher [1996], and Kolisch et al. [1999]).

In the original ProGen, Kolisch et al. [1992, 1995] also conducted several computational experiments that investigated the impact of the ProGen parameters used to characterize projects. The results demonstrate how ProGen parameters can indicate the degree of difficulty of a RCPSP. An analysis is conducted on several problem sets used as standard problem sets in previous research. Kolisch et al. [1992, 1995] found that these problem sets are a subset of easy problem instances. In addition, it was shown that hard instances can be far smaller in size than presumed in the literature and therefore may not be solved to optimality even with large amounts of computational time.

The standard ProGen developed by Kolisch et al. [1992, 1995] has three main parameters. For the purposes of this study, a modified ProGen, adapted from the standard ProGen, is developed. Two additional parameters are added in order to include resource vacations in the problem generation. A description of the modified ProGen is given in this section. A detail description of the standard ProGen can be found in Kolisch et al. [1992, 1995].

### 4.1.1 Task, Mode, and Duration Data Generation

The inputs for task, mode, and duration generation are: the minimum and maximum number of tasks in a project, $J_{\min}, J_{\max}$; the minimum and maximum number of executable modes for each task, $M_{j,\min}, M_{j,\max}$, and the minimum and maximum durations for each task-mode combination $d_{\min}, d_{\max}$. The duration for each task-mode combination is generated in this step through the use of uniformly distributed random

variables. The generated duration are then used to calculate the upper bound for the project completion time, $T$, which is the summation of the longest duration modes for each task,

$$T = \sum_{j=2}^{J-1} \max_{m=1}^{M_j} \{d_{j,m}\} \tag{4.1}$$

**Example 4.1** Suppose a project consists of 8 tasks. Each task can be processed in 2 possible modes. Each task-mode combination takes a duration ranging from 1 to 10 periods. With the given inputs, a duration for each task-mode combination is generated as shown in Table 4.1. Note that task index numbers for the 8-task project, after adding two dummy tasks (source and sink), are from 2 to 9. From the generated data, the upper bound for the project completion time is 63 periods.

Table 4.1 Duration for each task-mode combination

| Mode | Task | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 1 | 8 | 9 | 3 | 9 | 4 | 7 | 9 | 8 |
| 2 | 9 | 10 | 1 | 6 | 7 | 1 | 9 | 9 |

### 4.1.2 Network Generation

In general, the structure of a project can be depicted as either activity-on-arc (AOA) or activity-on-node (AON). The precedence relationships for projects in this research will be represented with AON networks. An AON network consists of a node set, $V$, and an arc set, $A$. For each node, $v \in V$, there exists a directed path from the single source (task 1)

to $v$ and a directed path from $v$ to the single sink (task $J$). This implies that every node except for the sink (source) has at least one successor (predecessor). A direct arc connecting two nodes $(a,b)$ may be redundant if there are arcs connecting $(a,i_1),(i_1,i_2),...,(i_{s-1},i_s),(i_s,b)\in A$. In order to capture network complexity, project instances considered in the MMRCPSP study must include only non-redundant arcs. A parameter used to characterize network structure in this step is called network complexity, $NC$, which is the average number of non-redundant arcs per node (including the dummy tasks). The construction of the network is performed in four steps.

Step 1: The numbers of *start* and *finish* tasks, $n_{sj}, n_{fj}$, are specified. Then, the arcs connecting the dummy source to the start tasks and the arcs connecting the finish tasks to the dummy sink are added to the network. Precedence arcs between the start tasks and finish tasks are not allowed.

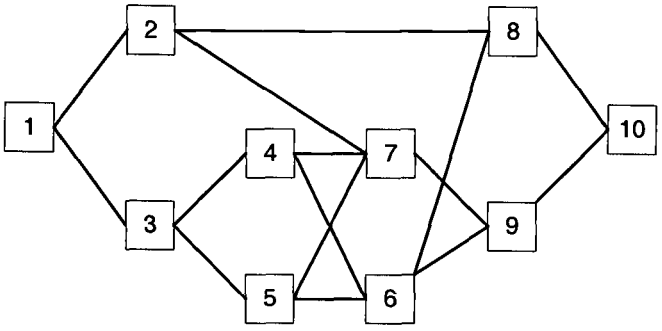
Step 2: Starting with the lowest indexed *non-start* tasks, each *non-finish* task is randomly assigned a predecessor that is connected to that task with an arc.

Step 3: Similarly, a successor is randomly assigned to each *non-finish* task yet to have a successor from step 2, starting with the lowest indexed non-start tasks.

Step 4: More arcs are added until the specified complexity is attained. Adding arcs in this step must avoid redundancy. In some cases, the generation procedure must be redone, since the randomly generated network does not allow the specified level of complexity.

**Example 4.1** (continued) Suppose the 8-task project has a network complexity of 1.5, a randomly generated network is shown in Figure 4.1.

Figure 4.1 A project with network complexity of 1.5




4.1.3 Resource Request Generation

The resources used by each task-mode combination must be generated. Kolisch et al. [1992, 1995] adopted the concept of resource factor, $RF$, from single-mode RCPSPs. The resource factor, $RF$, indicates the average fraction of the total number of different resources used per task. It is a normalized parameter (0-1 scale), where $RF = 0$ means that a task uses no resources, and $RF = 1$ means that each task uses some of each resource. For the multi-mode case, the $RF$ for each renewable resource, $r \in R$, is generalized as,

$$RF = \frac{1}{J-2}\frac{1}{|R|}\sum_{j=2}^{J-1}\frac{1}{M_j}\sum_{m=1}^{M_j} I(k_{j,m,r} > 0) \qquad (4.2)$$

**Example 4.1** (continued) Suppose there are two types of renewable resources available for the 8-task project. With a resource factor of 0.75, a resource usage for each task-mode could be randomly generated as shown in Table 4.2. For example, task 2 performed in mode 1 uses only resource type 1, whereas task 2 performed in mode 2 uses both resource types.

Table 4.2 Resource request by each task-mode combination

| Mode | Resource | Task | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| | 2 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 2 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |

### 4.1.4 Resource Requirements Generation – Resource Quantities

Given that a task-mode combination uses a set of resources, the resource demand, $k_{j,m,r}$, must be generated. The interrelationship between the task durations and the level of resource demand can be represented in two major ways. One is that the level of demand is independent of task duration. The other is that the total resource demand is constant over different duration (i.e. the level of demand is decreasing with increasing duration).

In this research task durations and resource demands are assumed to be independent. Under this assumption it is possible for some modes to be "inefficient". A mode is defined to be *inefficient* if there are modes $m$ and $m'$ with $d_{j,m} \leq d_{j,m'}$ and $k_{j,m,r} \leq k_{j,m',r}$ for all $r \in R$.

In standard ProGen, the resource demands are generated using a discrete uniform random variable in an interval $[U_{\min}, U_{max}]$. Total resource availability, $K_r$, is generated as a function of the level of demand associated with modes and a parameter called the resource strength, $RS$. Resource strength was first introduced by Cooper in 1976, and then used in ProGen. $RS$ is calculated as follows.

$$RS = \frac{K_r - K_r^{\min}}{K_r^{\max} - K_r^{\min}} \tag{4.3}$$

Resource strength is a measure of the "tightness" of resource levels. In ProGen $RS$ is normalized to a 0-1 scale. $RS = 0$ means that the resource availability for the project is at the minimum level such that the project still has a feasible schedule. $RS = 1$ means that the resource availability is sufficiently large such that resources do not constrain the schedule.

In order to generate $K_r$; $K_r^{\min}$ and $K_r^{\max}$ are calculated as follows.

$$K_r^{\min} = \max_{j=2}^{J-1} \left\{ \min_{m=1}^{M_j} (k_{j,m,r}) \right\} \tag{4.4}$$

$K_r^{\min}$ is the minimum resource level such that a project schedule is still feasible (i.e. the resource availability is so small that each task may be scheduled sequentially). The maximum capacity, $K_r^{\max}$, is calculated as the peak demand of the "precedence preserving earliest start schedule". In other words, each task is executed in a mode that requires the maximum level of demand, $k_{jr}^* = \max_{m=1}^{M_j} (k_{j,m,r})$, with the corresponding mode with shortest duration, $m_{j,r}^* = \min_{m=1}^{M_j} \left\{ d_{j,m} \mid k_{j,m,r} = k_{j,r}^* \right\}$.

Given the precedence relationships of a project, the earliest start schedule can be calculated with the predetermined modes. Thus, the peak period demand, $K_r^{\max}$, is calculated as,

$$K_r^{\max} = \max_{t=1}^{T} \left\{ \sum_{j=2}^{J-1} k_{j,m_{j,t},r} \right\} \tag{4.5}$$

With the values of $K_r^{\min}$ and $K_r^{\max}$ obtained, $K_r$ is calculated, with a user-specified $RS$, as,

$$K_r = K_r^{\min} + round(RS \times (K_r^{\max} - K_r^{\min})) \tag{4.6}$$

The ProGen procedure for generating level of demands, $k_{jmr}$, and resource availability, $K_r$, given above is not directly used. The reason is to limit the level of $K_r$ in order to reduce computational effort when solving problems. Therefore, instead the generation of resource demand in the modified ProGen is performed in reverse steps of the standard ProGen. Thus, generation of level of demand and resource availability in the modified ProGen involves the following steps.

Step 1: For each resource, calculate the maximum resource level required, $\left\{ K_r^{\max} \mid k_{j,m,r} \in [0,1] \right\}$, as in ProGen (all $k_{jmr}$ start with the binary values generated as in Section 4.1.3).

Step 2: With the values obtained in step 1, the resource availability, $K_r$, for each resource is set as an arbitrary value greater than $K_r^{\max}$.

Step 3: Given $K_r$, calculate the minimum upper bound for $k_{j,m,r}^{\min}$, that makes $K_r$ large for the project schedule.

$$k_{j,m,r}^{\min} = floor\left( \frac{K_r}{K_r^{\max} \mid k_{j,m,r} \in [0,1]} \right) \tag{4.7}$$

Step 4: $k_{j,m,r}$, is then generated from a discrete uniform distribution ranging from 1 to $k_{j,m,r}^{\max}$, where

$$k_{j,m,r}^{\max} = k_{j,m,r}^{\min} + round\left((1-RS)\times(K_r - k_{j,m,r}^{\min})\right) \tag{4.8}$$

The prior procedure still implements the concept of $RS$. With $RS = 0$, all $k_{j,m,r}$ are randomly drawn from $[1, K_r]$, which implies that each tasks resource demand can be as high as the maximum resource availability. With $RS = 1$, all $k_{j,m,r}$ are randomly drawn from the interval $[1, k_{j,m,r}^{\min}]$, which removes resource availability as a constraint when scheduling.

**Example 4.1** (continued) From the 0-1 data given in Table 4.2 and the precedence relationship given in Figure 4.1, there is a maximum of four tasks that can be scheduled at the same time (tasks (2,3), (4,5), (6,7), and (8,9)). Thus, $K_r^{\max}$ for both resource types is 2. In step 2, suppose one arbitrarily input $K_r$ to be twice as much as $K_r^{\max}$, then $K_r$ will take a value of 4 for both resource types.

In step 3, $k_{j,m,r}^{\min}$ is calculated to be $floor\left(\frac{K_r}{K_r^{\max} \mid k_{j,m,r} \in [0,1]}\right) = floor\left(\frac{4}{2}\right) = 2$. In step 4, if the resource strength is 0.75, then $k_{j,m,r}$ will be generated from a discrete uniform distribution ranging from 1 to $k_{j,m,r}^{\min} + round\left((1-RS)\times(K_r - k_{j,m,r}^{\min}\right) = 2 + round((1-0.75)\times(4-2)) = 3$. Therefore, for this example project, $k_{j,m,r}$ is a discrete uniformly distributed variable between 1 and 3 (Table 4.3).

Table 4.3 Resource demand by each task-mode combination

| Mode | Resource | Task | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 1 | 1 | 0 | 1 | 3 | 2 | 0 | 0 | 1 | 1 |
| | 2 | 3 | 1 | 0 | 0 | 1 | 3 | 0 | 2 |
| 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 3 | 1 |
| | 2 | 3 | 0 | 1 | 2 | 2 | 3 | 1 | 0 |

### 4.1.5 Resource Vacation Generation

Two additional parameters are defined to describe resource vacations: $V_P$ is the percent of time resources are not available due to vacations, and $V_L$ is the percent of vacations that are "short" in length. The percentage of time a resource is on vacation affects the resource capacity. The percent of short resource vacations reflects the variability of resource capacity. Shorter vacations result in higher resource capacity variability.

Generation of resource vacations involves three steps. First, the user inputs the percentage of resource vacations (i.e. a fraction of the planning horizon), $V_P$. From $V_P$, the total number of periods that each resource is on vacation is equal to $V_P T$, where $T$ is the upper bound of the project makespan obtained from (4.1).

Second, the percent of short vacations is specified. The total vacation periods are then broken down into *short* vacations, with a length generated from $U[d_{short}^{\min}, d_{short}^{\max}]$ random variables, and *long* vacations, generated from $U[d_{long}^{\min}, d_{long}^{\max}]$ random variables, according to $V_L$.

Finally, each resource vacation period is randomly placed throughout the project planning horizon, 1 to $T$. After inserting resource vacations on to the planning horizon in this step, the project planning horizon is adjusted by $T^* = (1 + p_{adj})T$ to assure feasibility of the problem instance, where $p_{adj} \in [0,1]$ is an arbitrary value.

**Example 4.1** (continued) The 8-task project generated thus far has 2 resource types, each with 4 units available ($K_r = 4, \forall r \in R$). If $V_P$ is 20%, then the total number of periods that every unit of each resource type is on vacation is $0.2 \times 63 \approx 13$. Also, suppose the percent of short vacations, $V_L$, is 50%, where short vacations are generated from $U[d_{short}^{min}, d_{short}^{max}] = U[1,2]$, and long vacations are generated from $U[d_{long}^{min}, d_{long}^{max}] = U[3,5]$. Then, the expected number of resource vacation periods for each resource unit will be 50% of short and 50% long. Once the vacations are generated and placed, a resource profile for each resource type with respect to time results, as shown in Figure 4.2.

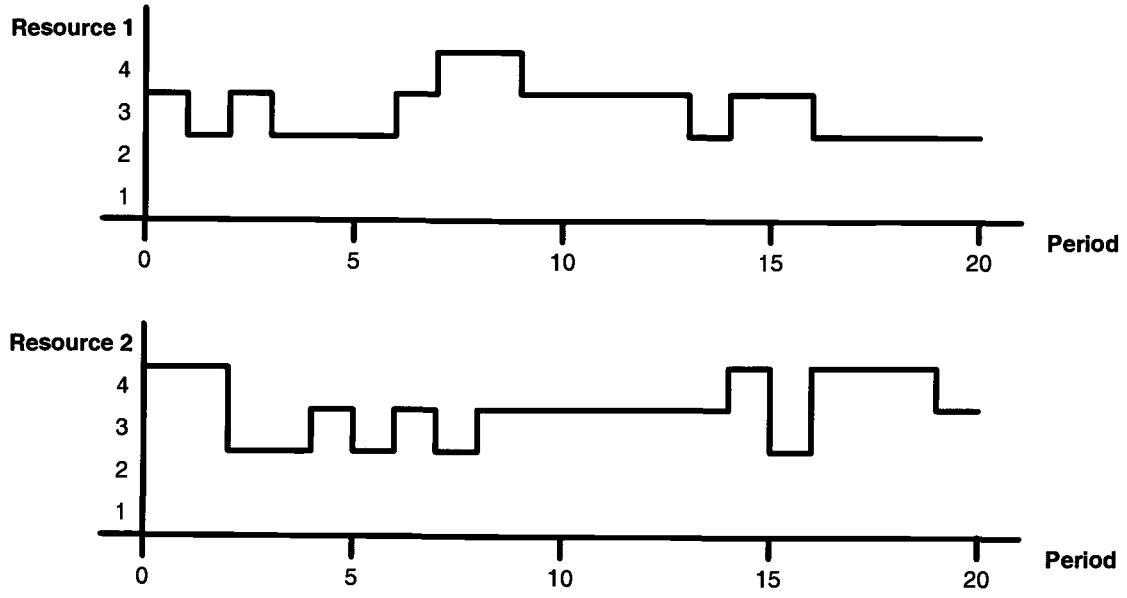



Figure 4.2 Resource profiles

## 4.2 EXACT ALGORITHM

This section describes the enumeration procedure used for solving small size MMRCPSPs in this research. The algorithm implemented here is a modification of the precedence tree algorithm, originally proposed by Patterson et al. [1989] and reconstructed by Hartman et al. [1998]. The algorithm is modified to handle task splitting and resource vacations. There are two versions of the modified precedence tree developed in this research. One schedules tasks without task splitting and the other with task splitting.

### 4.2.1 Enumeration Schemes

The formulation of the precedence tree algorithm starts with scheduling the dummy source task at time 0. At each level of the precedence tree, $g$, the set of the currently scheduled tasks $SJ_g$, and the set of the eligible activities $EJ_g$, are determined. Then an eligible task $j_g$, is selected and a mode $m_{j,g}$ for this task is selected. The next step is to determine the earliest precedence and resource feasible start time $s_{j,g}$. The algorithm continues branching the search tree until the dummy sink activity is eligible (i.e. a complete schedule is found). Then, the algorithm backtracks the search tree to the previous level and selects the next untested mode until all modes are tested. When all modes of the selected eligible task are tested, the algorithm selects the next untested eligible task. If all eligible tasks in all available modes are tested, the algorithm backtracks another step until all possible branches are examined (or pruned by bounding rules) and the optimal solution is found. The algorithm consists of the steps shown in Figure 4.3, where

$g$ := search tree level

$EJ_g$ := set of eligible tasks

$j_g$ := selected eligible task

$s_{j,g}$ := earliest precedence and resource feasible start time for task $j_g$.

$m_{j,g}$ := selected mode for task $j_g$

$M_{j,g}$ := the maximum number of modes for selected task $j_g$

$P_j$ := precedence relationship for task $j_g$

$SJ_g$ := set of currently scheduled tasks

$UM_g$ := set of untested modes at level $g$

**Example 4.1** (continued) Using the modified precedence tree algorithm to solve this project instance, the optimal schedule is as illustrated in Figure 4.4. The optimal makespan for the project without task splitting is 34 periods, whereas the optimal makespan for the project with task splitting is 29 periods. Thus, task splitting for this particular project yields an improvement in makespan of $\frac{34-29}{34} \times 100 = 14.70\%$.

### 4.2.2 Bounding Rules

Four bounding criteria are applied to speed up the enumeration procedure. These rules are found in the literature. Some of the rules implemented here are adjusted to fit the framework for task splitting.

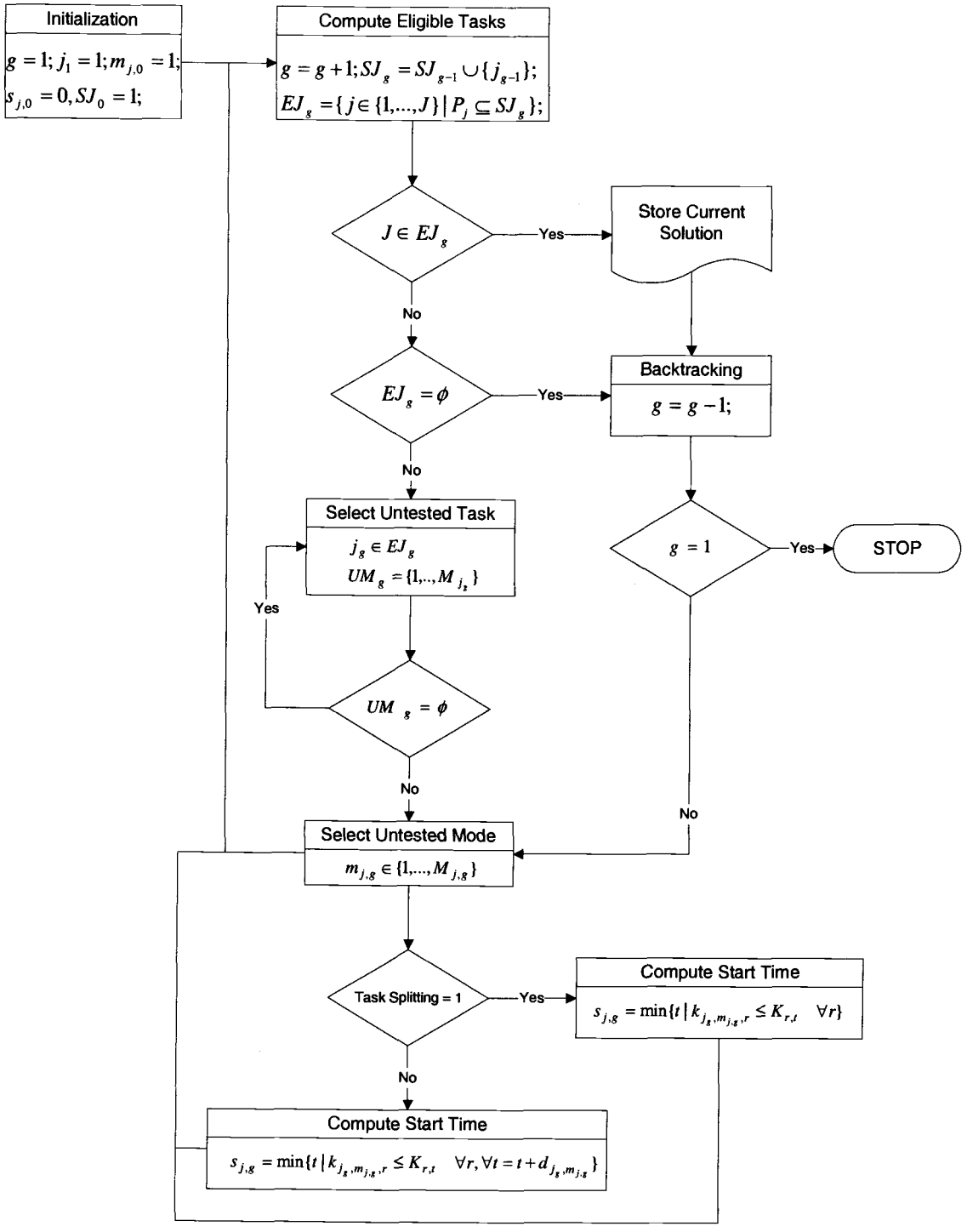



Figure 4.3 Precedence tree algorithm

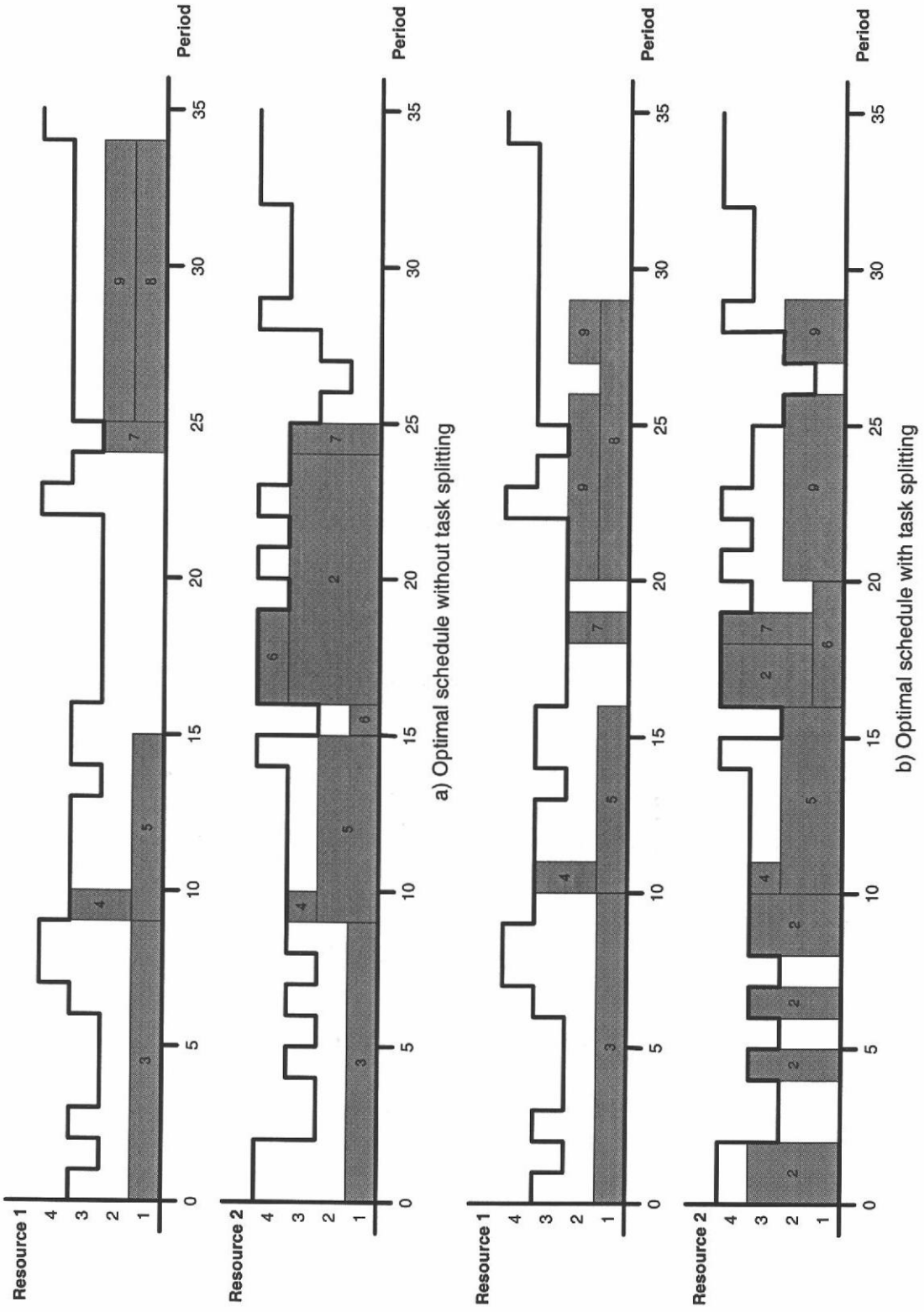



Figure 4.4 Optimal schedules for Example 4.1

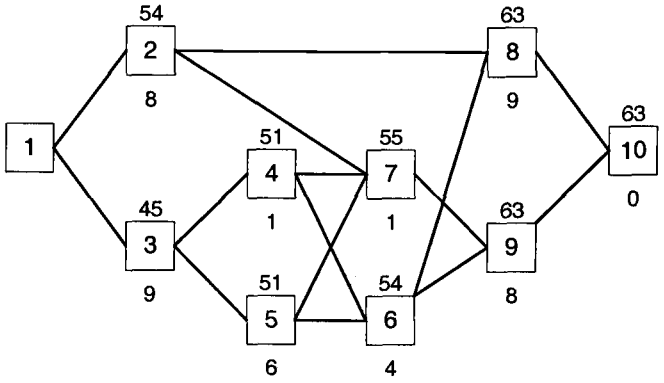
#### 4.2.2.1 Bounding Rule 1: Data Reduction by Eliminating all Inefficient Modes

This rule is applied as a preprocessor. It was originally proposed by Sprecher et al. [35]. As explained in Section 4.1.4, a mode is defined as *inefficient* if its duration is not shorter and its resource demands are not less than those of another mode of the same task. Thus, before solving a problem instance, inefficient modes may be excluded from the project data by preprocessing, without loss of optimality. For instance, consider the resource demand in Example 4.1. Modes 2 for task 2, task 6, and task 9 are inefficient and can be excluded from the project data

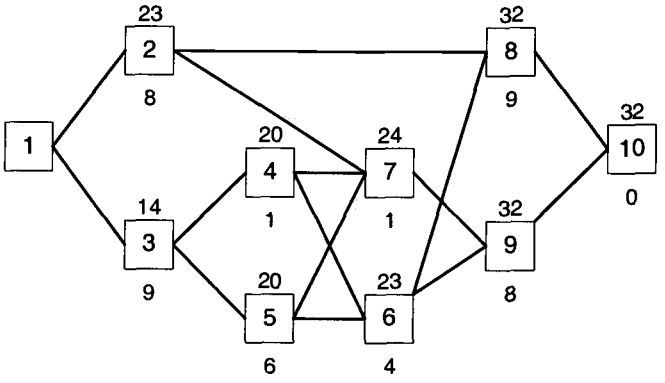
#### 4.2.2.2 Bounding Rule 2: Time Window Rule

Let $T$ denote an upper bound on project makespan. If there is a scheduled task $j$ with an assigned finish time that exceeds its latest finish time $LFT_j$, then the current partial schedule need not be completed, since it cannot be completed with a better makespan than $T$.

This bounding rule prunes the search tree as follows. The enumeration scheme begins with an initial upper bound calculated in (4.1), and successively updates it when the first solution or an improved solution is found. Given the precedence relations and an initial upper bound on the project, the latest finish time $LFT_j$ for each task can be derived by selecting the modes of shortest duration and perform a traditional backward recursion as in *CPM* .




(a) Initial latest finish time




(b) Adjusted latest finish time

Figure 4.5 Latest finish time for each task

Let $T$ denote an initial upper bound of the project. The initial $LFT_j$ for each task $j$ can be calculated recursively (starting with task $J-1$), as the minimum, among all task $j'$s succeeding $j$, of the finish time minus the minimum duration of task $j$: $LFT_j = \min_{j'}(FT\{j' \mid j \rightarrow j'\} - \min_{m}(d_{j,m}))$. Let $T$ denote a newly obtained upper bound from a complete schedule. Then the latest finish time $LFT_j$ can be recalculated for new values of $T$ by $LFT_j = LFT_j - (LFT_J - T + 1)$.

**Example 4.1** (continued) The project instance has the initial upper bound of the makespan $T = \sum_{j=2}^{J-1} \max_{m=1}^{M_j} \{d_{j,m}\} = 63$. With the precedence relations given in Figure 4.1, the initial $LFT_j$ for each task $j$ is given in Figure 4.5 (a), where the number above each task index is its initial $LFT_j$, and the number below is its minimum duration. If an improved makespan is found to be 33, the latest finish time will be adjusted as in Figure 4.5 (b).

#### 4.2.2.3 Bounding Rule 3: Non-delayability

If an eligible task cannot be feasibly scheduled in any mode in the current partial schedule without exceeding its latest finish time, then there is no need to examine other eligible tasks on this search tree level.

Suppose at a search level $g$, there are two eligible tasks, $EJ_g = \{j, j'\}$. If task $j$ exceeds its latest finish time if scheduled, while task $j'$ can be selected without violating Bounding rule 2, then, at lower levels in the same search tree task $j$ will have to be scheduled and will have a finish time exceeding its latest finish time. Therefore, applying Bounding rule 3 can prune this search tree earlier.

#### 4.2.2.4 Bounding Rule 4: Precedence Tree Rule

Consider two tasks $j$ and $j'$ scheduled on consecutive levels of the same branch. Suppose, at level $g-1$ task $j$ is selected and at level $g$ task $j'$ is selected, resulting in start times $s_j$ and $s_{j'}$ for task $j$ and task $j'$, respectively. Then, suppose task $j'$ is instead selected at level $g-1$ and task $j$ at level $g$, resulting in $s_{j'}$ and $s_j$. If the start

times for both tasks are equal in both cases, then examining both branches will yield the same result. Therefore, one of these branches need not be completed.

## 4.3 EXPERIMENT 1: INITIAL FACTOR SCREENING

The objective of experiment 1 is to identify project situations where task splitting is likely to improve a schedule. Such project situations are identified by analyzing which modified ProGen parameters significantly affect the schedule improvement when task splitting is allowed. Another intent of this experiment is to identify unimportant ProGen parameters relative to their use in identifying when task splitting is beneficial.

### 4.3.1 Experimental Design

In the modified ProGen, there are five factors parameterized to generate project instances. To limit the number of runs, each factor is examined at 2 levels. In addition, each project must be solved with and without task splitting. Therefore, this experiment is set up as a $2^5$ full factorial design. The required base data used to generate small size project instances are given in Table 4.4. The two levels for each factor are shown in Table 4.5.

### 4.3.2 Response Variable

The response variable in this experiment is a binary variable that indicates whether or not there is an improvement of the optimal solution when task splitting is permitted. The percent improvement in optimal makespan is defined as,

Table 4.4 Base data for generating small size problems

| Base data | Minimum | Maximum |
|---|---|---|
| Number of tasks (experiment 1) | 10 | 10 |
| Number of tasks (experiment 2 and 3) | 8 | 13 |
| Modes per task | 2 | 2 |
| Task duration | 1 | 10 |
| Number of resources | 2 | 2 |
| Number of resource requested | 1 | 2 |
| Number of task 1 (dummy start) successors | 3 | 3 |
| Number of task $j$ successors | 1 | 3 |
| Number of predecessors of task $J$ (dummy finish) | 3 | 3 |
| Number of predecessors of task $j$ | 1 | 3 |
| Short vacation length | 1 | 2 |
| Long vacation length | 3 | 5 |

Table 4.5 Levels of experimental factors

| Experimental factors | Level 1 | Level 2 |
|---|---|---|
| Network complexity, $NC$ | 1.5 | 1.8 |
| Resource factor, $RF$ | 0.5 | 1.0 |
| Resource strength, $RS$ | 0.25 | 0.75 |
| Percent resource vacation, $V_P$ | 10% | 20% |
| Resource vacation length, $V_L$ | short | long |

$$\%\ \text{improvement} = 100 \times \frac{\text{makespan(w/o task splitting)} - \text{makespan(with task splitting)}}{\text{makespan(w/o task splitting)}} \quad (4.9)$$

For this experiment with each problem instance solved with and without task splitting, a total of $2 \times 2^5 \times 20 = 1{,}280$ runs are required. Each response is calculated from a pair of runs on the same problem resulting in 640 observations of percent improvement.

Table 4.6 Summary of results for each factor combination

| Factor combination | *NC* | *RF* | *RS* | $V_P$ | $V_L$ | count (out of 20) | Improvement proportion |
|---|---|---|---|---|---|---|---|
| 1 | 1.5 | 0.5 | 0.25 | 10% | short | 10 | 0.50 |
| 2 | 1.5 | 0.5 | 0.25 | 10% | long | 4 | 0.20 |
| 3 | 1.5 | 0.5 | 0.25 | 20% | short | 19 | 0.95 |
| 4 | 1.5 | 0.5 | 0.25 | 20% | long | 10 | 0.50 |
| 5 | 1.5 | 0.5 | 0.75 | 10% | short | 1 | 0.05 |
| 6 | 1.5 | 0.5 | 0.75 | 10% | long | 0 | 0 |
| 7 | 1.5 | 0.5 | 0.75 | 20% | short | 11 | 0.55 |
| 8 | 1.5 | 0.5 | 0.75 | 20% | long | 5 | 0.25 |
| 9 | 1.5 | 1 | 0.25 | 10% | short | 18 | 0.90 |
| 10 | 1.5 | 1 | 0.25 | 10% | long | 11 | 0.55 |
| 11 | 1.5 | 1 | 0.25 | 20% | short | 20 | 1.0 |
| 12 | 1.5 | 1 | 0.25 | 20% | long | 16 | 0.80 |
| 13 | 1.5 | 1 | 0.75 | 10% | short | 10 | 0.50 |
| 14 | 1.5 | 1 | 0.75 | 10% | long | 4 | 0.20 |
| 15 | 1.5 | 1 | 0.75 | 20% | short | 14 | 0.70 |
| 16 | 1.5 | 1 | 0.75 | 20% | long | 11 | 0.55 |
| 17 | 1.8 | 0.5 | 0.25 | 10% | short | 10 | 0.50 |
| 18 | 1.8 | 0.5 | 0.25 | 10% | long | 5 | 0.25 |
| 19 | 1.8 | 0.5 | 0.25 | 20% | short | 17 | 0.85 |
| 20 | 1.8 | 0.5 | 0.25 | 20% | long | 9 | 0.45 |
| 21 | 1.8 | 0.5 | 0.75 | 10% | short | 3 | 0.15 |
| 22 | 1.8 | 0.5 | 0.75 | 10% | long | 1 | 0.05 |
| 23 | 1.8 | 0.5 | 0.75 | 20% | short | 8 | 0.40 |
| 24 | 1.8 | 0.5 | 0.75 | 20% | long | 5 | 0.25 |
| 25 | 1.8 | 1 | 0.25 | 10% | short | 18 | 0.90 |
| 26 | 1.8 | 1 | 0.25 | 10% | long | 9 | 0.45 |
| 27 | 1.8 | 1 | 0.25 | 20% | short | 20 | 1.0 |
| 28 | 1.8 | 1 | 0.25 | 20% | long | 18 | 0.90 |
| 29 | 1.8 | 1 | 0.75 | 10% | short | 6 | 0.30 |
| 30 | 1.8 | 1 | 0.75 | 10% | long | 2 | 0.10 |
| 31 | 1.8 | 1 | 0.75 | 20% | short | 12 | 0.60 |
| 32 | 1.8 | 1 | 0.75 | 20% | long | 8 | 0.40 |

4.3.3 Experimental Results and Statistical Analysis

The experiment results are summarized for each of the 32 factor combinations in Table 4.6. The overall proportion of improvement is $\frac{315}{640} = 0.49$, with a standard deviation of 0.30. The high variability of proportion of improvement is also indicated in Table 4.6, where some factor combinations resulted in little improvement, and some combinations produced very large improvements. Table 4.7 summarizes the proportion of improvement for each factor separately to help visualize the effect of each factor.

Table 4.7 Summary of results for each factor

| Factor | *NC* | | *RF* | | *RS* | | $V_P$ | | $V_L$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| Level | 1.5 | 1.8 | 0.5 | 1.0 | 0.25 | 0.75 | 10% | 20% | short | long |
| Count | 164 | 151 | 118 | 197 | 214 | 101 | 112 | 203 | 197 | 118 |
| Proportion | 0.51 | 0.47 | 0.37 | 0.62 | 0.67 | 0.32 | 0.35 | 0.63 | 0.62 | 0.37 |

In order to determine what factors and 2-way interactions of factors are significant in producing improvement on the solutions, the binary responses are used, where 0 indicates no improvement and 1 indicates improvement on the solution. With binary responses, a logistic regression analysis (McCullagh and Nelder [1989]) is performed (details of the statistical analysis can be found in Appendix A).

A logistic regression model containing all main factors and all 2-way interactions is fitted. It is important to note that all higher order interactions are not of interest and therefore their effects are included in the experimental error terms. Table 4.8 provides the p-values from the model (p-values of significant factors are highlighted in bold). The logistic regression model is,

Table 4.8 Significance level for each experimental factor

| Experimental factor | p-value |
|---|---|
| $NC$ | 0.2125 |
| $RF$ | **< 0.0001** |
| $RS$ | **< 0.0001** |
| $V_P$ | **< 0.0001** |
| $V_L$ | **< 0.0001** |
| $NC \times RF$ | 0.3594 |
| $NC \times RS$ | 0.7208 |
| $NC \times V_P$ | 0.6108 |
| $NC \times V_L$ | 0.5211 |
| $RF \times RS$ | 0.1754 |
| $RF \times V_P$ | 0.4766 |
| $RF \times V_L$ | 0.5621 |
| $RS \times V_P$ | 0.9263 |
| $RS \times V_L$ | **0.0223** |
| $V_P \times V_L$ | 0.8040 |

$$\log\left(\frac{\pi_i}{1-\pi_i}\right) = 0.1391 - 0.0063NC + 2.2890RF - 2.1569RS + 2.0256V_P$$
$$-1.9748V_L - 0.3795(NC \times RF) - 0.1527(NC \times RS) - 0.2138(NC \times V_P) \qquad (4.10)$$
$$+0.2663(NC \times V_L) - 0.6122(RF \times RS) - 0.3138(RF \times V_P) - 0.2550(RF \times V_L)$$
$$-0.0420(RS \times V_P) + 1.0348(RS \times V_L) - 0.1102(V_P \times V_L)$$

where $NC$, $RF$, $RS$, $V_P$, and $V_L$ are all binary indicator variables, which take a value of 1 when the factor is at level 2 (according to Table 4.5), and 0 otherwise.

### 4.3.4 Interpretations of the Results

Results from the logistic regression models are normally interpreted in terms of an odds ratio. In this research the statistical statements are made in the context of MMRCPSPs and the experimental factors used in the modified ProGen (more details of the statistical statements are presented in Appendix A). To estimate the effect of each significant experimental factor, a reduced model that contains only significant factors is constructed using a model selection process. The reduced model, given in (4.11), is used to estimate the effect of each factor in terms of an odds ratio.

$$\log\left(\frac{\pi_i}{1-\pi_i}\right) = 0.36 + 1.46RF - 2.40RS + 1.66V_P - 1.85V_L + 0.79(RS \times V_L) \qquad (4.11)$$

Network complexity: There is no statistical evidence that the variation between two levels of network complexity has a significant effect on the probability of schedule improvements with task splitting. After accounting for $RF$, $V_P$, $RS$, $V_L$, and $RS \times V_L$, p-value = 0.2125.

Resource factor: There is convincing evidence that with a resource factor of 1.0 task splitting has a better chance of improving a schedule than with a resource factor of 0.5. After accounting for $RS$, $V_P$, $V_L$, and $RS \times V_L$, p-value < .0001. The odds of improving the solution for an instance with $RF$ of 1.0 were estimated to be 4.29 times the odds of improving the solution for an instance with $RF$ of 0.5, when holding all other factors fixed.

Percent resource vacation: After accounting for the other factors, there is convincing evidence that for a project with 20% resource vacations, task splitting has more chance of improving the schedule than with 10% resource vacations, p-value < .0001. Holding the values of the other factors fixed, the odds of improving the schedule for a project with

20% resource vacations were estimated to be 5.25 times the odds of improving the schedule of a project with 10% resource vacations.

The interpretation of $RS$ and resource vacation length $V_L$, must be made together because the interaction between resource strength and the length of resource vacations is significant.

Resource strength: After taking the other factors into account, when resources are more constrained task splitting has a higher chance of improving a schedule, p-value = 0.0223. For a project with long (short) resource vacation lengths, the odds of task splitting improving the solution when $RS$ is 0.25 is estimated to be 10.99 (31.72) times that when $RS$ is 0.75 (when holding the other factors fixed).

Resource vacation length: After adjusting for the other factors, there is suggestive evidence that in a project with short resource vacation lengths, task splitting has a better chance improving a schedule than when resource vacation lengths are long, p-value = 0.0223. At a $RS$ of 0.25 (0.75), the odds of task splitting improving a schedule for a project with short vacations is estimated to be 6.36 (31.72) times that when the project has long vacations (when holding the other factors constant).

Rearranging the summarized results for $RS$ and $V_L$ in Table 4.7, it can be seen in Table 4.9 that the interpretations for these two effects are consistent with the observations from the experiment.

Table 4.9 Marginal summary of results for $RS$ and $V_L$

| | | $V_L$ | |
|---|---|---|---|
| | | short | long |
| $RS$ | 0.25 | 132<br>82.50% | 82<br>51.25% |
| | 0.75 | 65<br>40.63% | 36<br>22.50% |

## 4.4 EXPERIMENT 2: MAGNITUDE OF SOLUTION IMPROVEMENT

Experiment 2 is an investigation of the effects of the factors on the degree of schedule improvement, given that task splitting does improve a schedule.

### 4.4.1 Experimental Design

Experiment 2 includes all four significant factors from experiment 1, ($RF$, $V_P$, $RS$, and $V_L$). The experiment is a $2\times3\times4\times3\times3$ full factorial design, with the levels of factors added as shown in Table 4.10. In addition, the scope of inference of the study is extended by adding more data on the number of tasks included in a project, as shown in Table 4.4.

### 4.4.2 Response Variable

For this experiment, the goal is to investigate the factor effects on the degree of solution improvement. The responses, given in (4.9), are used in the analysis. A total of

$2\times3\times4\times3\times3\times5\times6=6,480$ projects are optimally scheduled. This yields 3,240 observations of percent improvement in schedule.

Table 4.10 Levels of experimental factors

| Experimental factors | Level 1 | Level 2 | Level 3 | Level 4 |
|---|---|---|---|---|
| Resource factor, *RF* | 0.5 | 0.75 | 1.0 | |
| Resource strength, *RS* | 0.25 | 0.50 | 0.75 | 1.0 |
| Percent resource vacation, $V_P$ | 10% | 20% | 30% | |
| Resource vacation length, $V_L$ | short | medium[1] | long | |

Note: [1] medium resource vacation length indicates that resource vacations are generated as 50% short vacations and 50% long vacations.

### 4.4.3 Experimental Results and Statistical Analysis

From a total of 3,240 observations, 1,685 showed improvement, giving a proportion of improvement of 0.52. Because the responses are numerical values, multiple linear regression is performed with the log-transformed percent improvement as a response, and the four main factors and the number of tasks as categorical explanatory variables. The response is transformed to satisfy the statistical model assumptions (validation of model assumptions is covered in Appendix B).

Similar to the analysis in Section 4.3.3, the model with all main effects and the number of tasks as well as all 2-way interactions was fit resulting in the significance levels in Table 4.11. A reduced model was fit to estimate the effect of each significant factor (equation 4.12).

$$\begin{aligned}\log(\%improvement) &= 2.32 - 0.05TASK - 0.16RS_{0.5} - 0.14RS_{0.75} \\ &- 0.29RS_{1.0} + 0.55V_{P,20\%} + 1.07V_{P,30\%} + 0.26V_{L,medium} + 0.40V_{L,short} \\ &- 0.24RS_{0.5} \times V_{P,20\%} - 0.36RS_{0.5} \times V_{P,30\%} - 0.38RS_{0.75} \times V_{P,20\%} \\ &- 0.56RS_{0.75} \times V_{P,30\%} - 0.42RS_{1.0} \times V_{P,20\%} - 0.57RS_{1.0} \times V_{P,30\%}\end{aligned} \quad (4.12)$$

where $RS$, $V_P$, and $V_L$ are all binary indicator variables, which take a value of 1 when the factor is equal to its subscript (Table 4.10), and 0 otherwise. The effects of each factor on the percent improvement estimated in (4.12) are linear on a log-scale. When back-transformed to the original scale, the factor effects are multiplicative. For example, the additive coefficient of $RS_{0.5}$ is -0.16 on the log scale of percent improvement. After back-transformation, $RS_{0.5}$ has $e^{-0.16} = 0.85$ multiplicative effect on the percent improvement.

Table 4.11 Significance level of experimental factors

| Experimental factors | p-value |
|---|---|
| $TASK$ | **<.0001** |
| $RF$ | 0.5271 |
| $RS$ | **<.0001** |
| $V_P$ | **<.0001** |
| $V_L$ | **<.0001** |
| $TASK \times RF$ | 0.6153 |
| $TASK \times RS$ | 0.7540 |
| $TASK \times V_P$ | 0.9129 |
| $TASK \times V_L$ | 0.2962 |
| $RF \times RS$ | 0.4614 |
| $RF \times V_P$ | 0.5405 |
| $RF \times V_L$ | 0.6057 |
| $RS \times V_P$ | **<.0001** |
| $RS \times V_L$ | 0.0699 |
| $V_P \times V_L$ | 0.0373 |

### 4.4.4 Interpretations of the Results

The results from Table 4.11 and the final model (4.12) indicate significance of three main effects, the number of tasks, and an interaction term. According to the p-values in Table 4.11, the different levels of $RF$ have no effect on the degree of improvement resulting from task splitting. Thus, it was eliminated from the final model. The number of tasks is found to be significant. Therefore, it is retained in the model so that its effect is accounted for in the statistical analysis for the other factors in the reduced model. The main factor, $V_L$, is significant and does not interact with other factors.

Due to the significance of their interaction, $RS$ and $V_P$ cannot be interpreted separately. To cope with the interaction of factors with more than two levels, orthogonal contrasts are used to break the levels of each factor into an orthogonal mean effect, a linear effect, a quadratic effect, and possibly a cubic effect (only for the factor with 4 levels). The $RS \times V_P$ interactions are broken down into six orthogonal interactions. The level of significance of each contrast is estimated and shown in Table 4.12.

Table 4.12 Level of significance for the orthogonal contrasts

| Orthogonal interaction | p-value |
|---|---|
| $V_{L,linear}$ | **<.0001** |
| $V_{L,quadratic}$ | 0.0992 |
| $RS_{linear} \times V_{P,linear}$ | **0.005** |
| $RS_{quadratic} \times V_{P,linear}$ | 0.1786 |
| $RS_{cubic} \times V_{P,linear}$ | 0.9348 |
| $RS_{linear} \times V_{P,quadratic}$ | 0.3958 |
| $RS_{quadratic} \times V_{P,quadratic}$ | 0.8713 |
| $RS_{cubic} \times V_{P,quadratic}$ | 0.9222 |

The effects are illustrated in Figure 4.6, Figure 4.7, and Figure 4.8 to help understand their meaning. Figure 4.6 shows the *linear* effect of $V_L$ on the median of the percent improvement. Figure 4.7 and Figure 4.8 reveal the relationship between $RS$ and $V_P$ on the median of the percent improvement. The significance of $RS \times V_P$ interaction lies in the $RS_{linear} \times V_{P,linear}$ term.

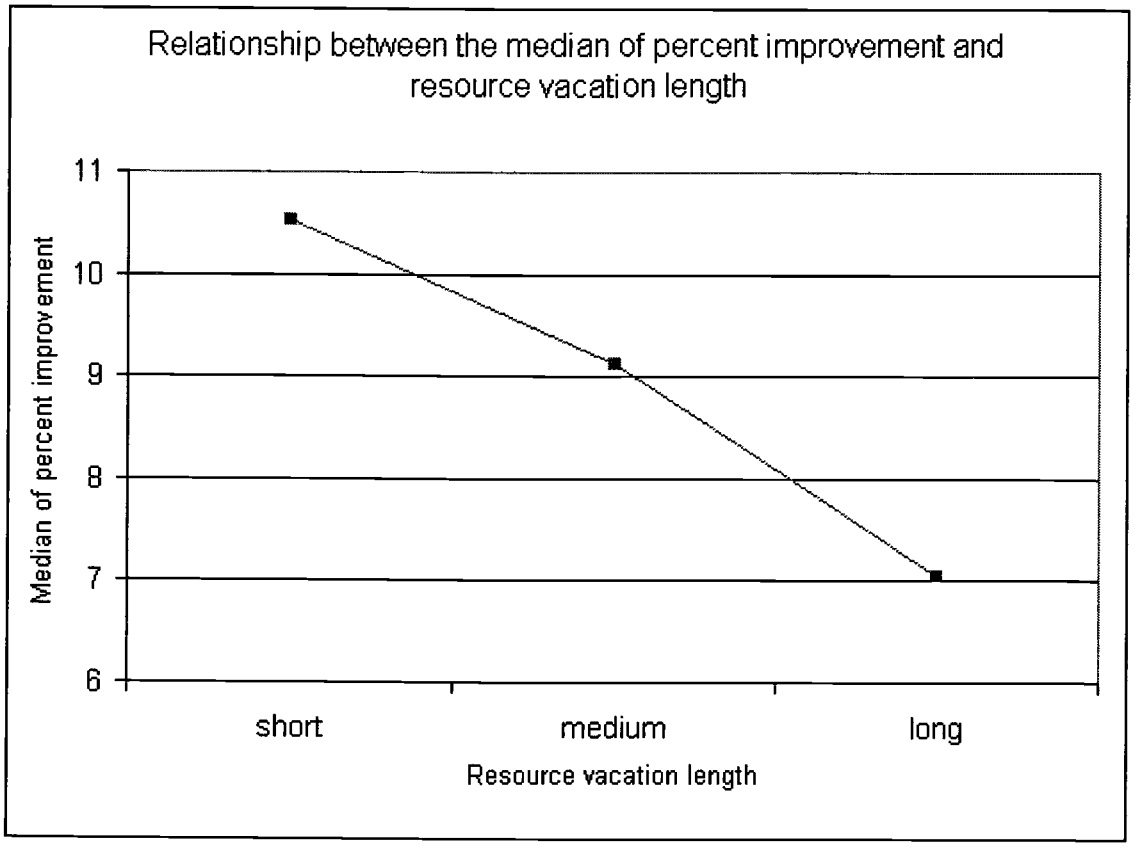



Figure 4.6 Effect of resource vacation length

Resource vacation length: The medians of the percent improvement due to task splitting increase as the length of resource vacations decrease. Changing the resource vacation

length from long to medium (from long to short) increases the median of percent improvement by 29.69% (49.18%), with a 95% confidence interval of (19.52%, 40.73%), after adjusting for $TASK$, $V_P$, and $RS$, (49.18% with a 95% confidence interval of (37.37%,62.02%) for short vacations).

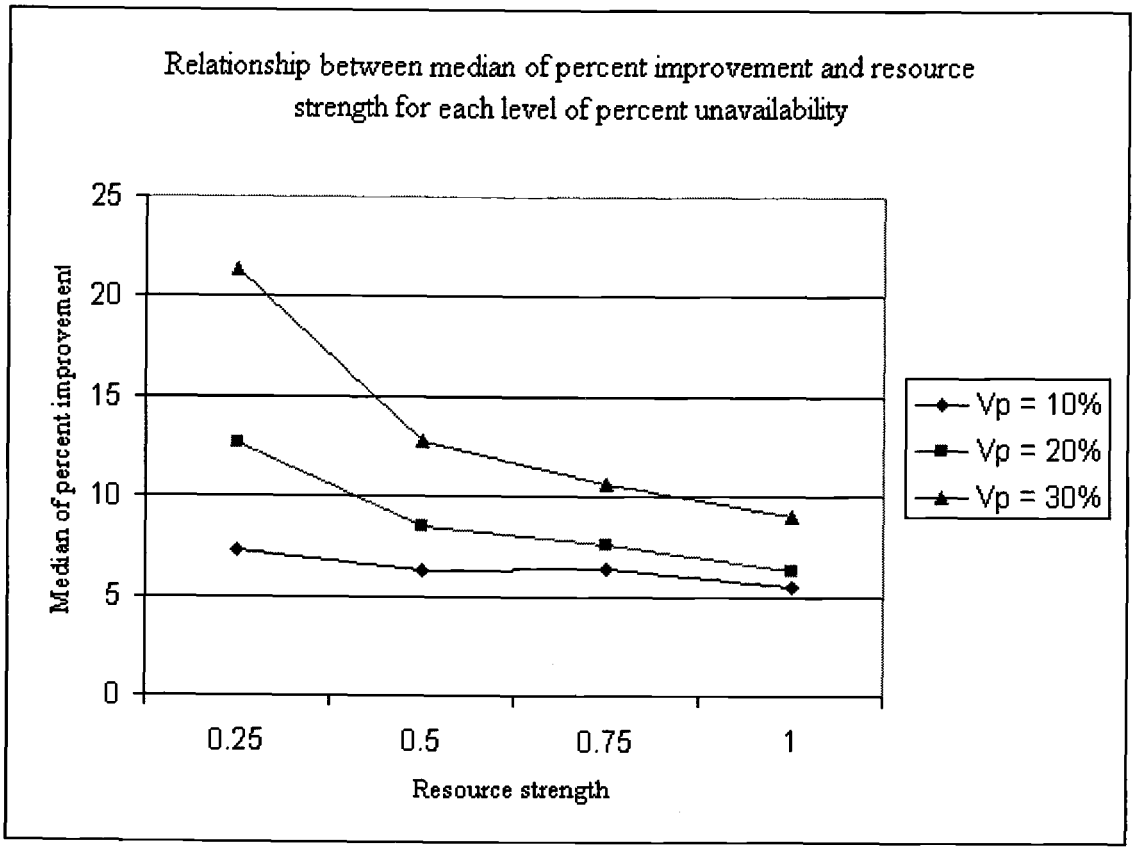



Figure 4.7 Effects of $RS$ on the degree of improvement of the solutions

Resource strength: The median percent improvement in makespan due to task splitting decrease as resource constraints are loosened (i.e. increasing values of $RS$). In addition,

as the percent resource vacation increases, the medians of the percent improvement decrease more under looser resource constraints. In Figure 4.8, the slopes of the relationships between the medians percent improvement and $RS$ tend to be steeper as the percent resource vacation increases.

Changing the level of $RS$ from 0.25 to other levels will decrease the median percent improvement in makespan due to task splitting, depending on the level of percent resource vacation. Table 4.13 summarizes the percent decrease in the median percent improvement for each combination of resource strength and percent resource vacation. The estimated 95% confidence intervals are also reported in Table 4.13.

Table 4.13 Effects of resource strength as percent resource vacation varies

| Each cell is a decrease in median of percent improvement by | | changing $RS$ from 0.25 to | | |
|---|---|---|---|---|
| | | 0.50 | 0.75 | 1.0 |
| at $V_P$ of | 20% | 32.97% (23.45%,41.31%) | 40.55% (31.37%,48.50%) | 50.84% (37.88%,61.09%) |
| | 30% | 40.55% (32.89%,47.33%) | 50.34% (43.71%,56.19%) | 57.68% (50.27%,63.99%) |

Percent resource vacation: The medians percent improvement in makespan due to task splitting increase as resource availability decreases (i.e. increasing values of $V_P$). These increments depend on the tightness of resource constraints (i.e. the value of $RS$). As resource constraints become tighter, the median percent improvements increase more as the resource availabilities decrease. Figures 4.9 shows that the slopes of the relationships between the median percent improvements and $V_P$ are steeper as $RS$ decreases.

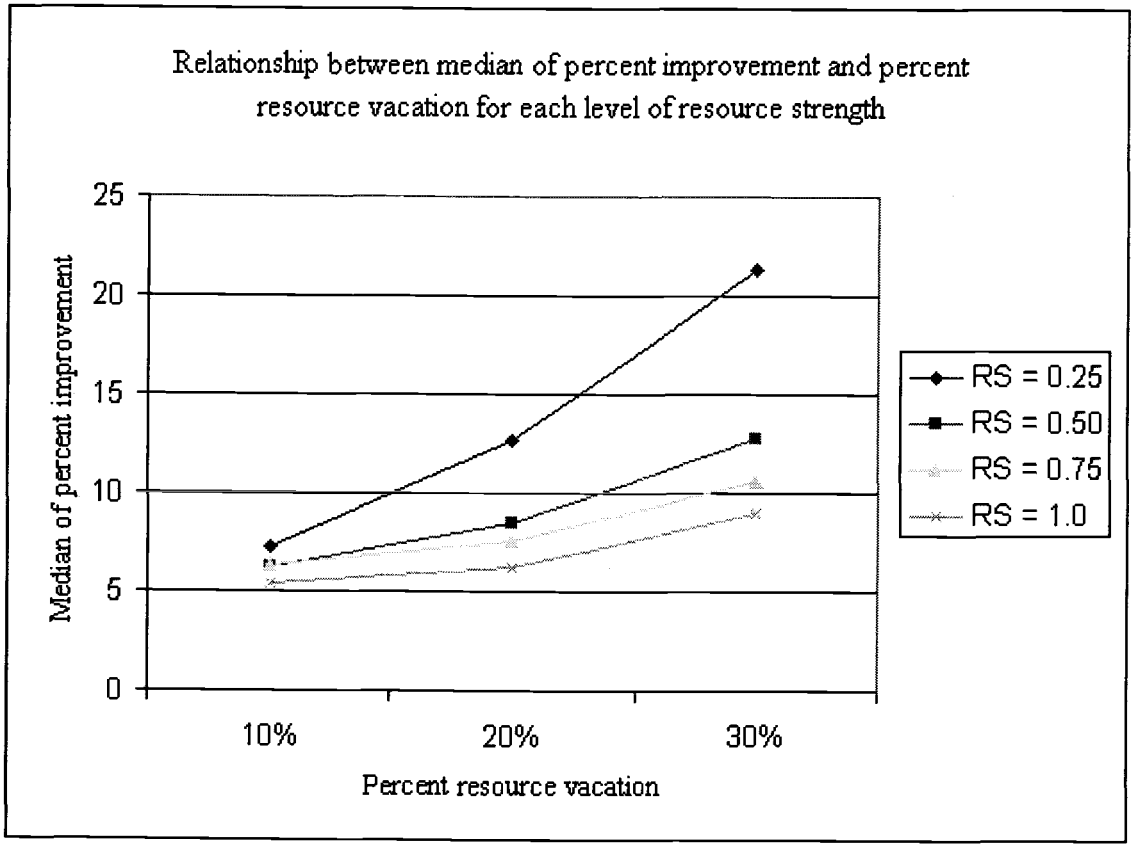



Figure 4.8 Effects of $V_P$ on the degree of improvement of the solutions

Changing $V_P$ from 10% to other levels will increase the median percent improvement, depending on the level of resource strength. Table 4.14 summarizes the percent increase in the median percent improvement for each combination of resource strength and percent resource vacation, as well as the associated 95% confidence intervals. From the confidence interval shown, some effects are inconclusive due to the confidence intervals including zero.

Table 4.14 Effects of percent resource vacation as resource strength varies

| Each cell is a increase in median of percent improvement by | | changing $V_P$ from 10% to | |
|---|---|---|---|
| | | 20% | 30% |
| at *RS* of | 0.50 | 36.34% (16.48%,59.60%) | 103.40% (74.87%,136.58%) |
| | 0.75 | 18.53% (-1.45%,42.56%) | 66.53% (40.00%,98.08%) |
| | 1.0 | 13.88% (-27.53%,78.96%) | 64.87% (8.30%,150.99%) |

4.4.5 Conclusions from Experiment 1 and Experiment 2

The computational experiments on small size problems conducted thus far lead to some insights. The project parameters that have significant effects on whether task splitting improves a schedule are: resource factor, resource strength, percent resource vacation, and length of resource vacations. Investigation on the levels of these parameters suggest that task splitting has a better chance of improving a schedule when a project is more difficult to schedule (i.e. high resource factor, tight resource constraints, high percent of resource vacation, and high variability of resource availability due to shorter length of vacation).

If a project schedule is improved by task splitting, the project parameters that have significant effects on the degree of improvement are resource strength, percent resource vacation, and length of vacation period. All three significant factors are directly connected to the availability of resources (reflected by resource strength and percent vacation) as well as its variation (parameterized as vacation length).

The scope of inference for the results obtained here is limited to the MMRCPSPs that have the number of tasks and values of project parameters used in these experiments. Making inferences beyond the scope of the experiments is speculative.

# 5. HEURISTIC METHODOLOGY FOR MEDIUM AND LARGE PROBLEMS

This chapter presents the heuristic methodology developed for scheduling medium to large projects. The objectives of this heuristic are: to generate good schedules for projects that are too large to solve optimally; and to realize the benefits of task splitting with the smallest number of splits possible. Section 5.1 describes the components of the heuristic as well as a new concept introduced to control task splitting. Section 5.2 presents the results from applying the heuristic on the small size problems with known optimal solutions. Section 5.3 demonstrates the use of the heuristic scheduling medium and large projects.

## 5.1 DESCRIPTION OF THE HEURISTIC

This section describes the priority rule-based heuristic methods used in this research for solving the large MMRCPSPs. This heuristic consist of three components, including a schedule generation scheme (SGS), a set of priority rules, and a selection methods. A schedule generation scheme is a procedure that generates feasible schedules by augmenting partial schedules in a step-wise manner. In each step, the SGS determines the set of all eligible task-mode options that may be scheduled. A set of priority rules is used to evaluate the priority value for each eligible task-mode option. A task selection method, based on the priority values, is then applied to select a task-mode option to schedule. A new concept called the "*moving resource strength*" is developed to control task splitting that is permitted in the heuristic.

### 5.1.1 Schedule Generation Schemes

Schedule generation schemes are procedures that build complete schedules by stepwise extension of a partial schedule. A partial schedule is a schedule where only a subset of the $J$ tasks (including two dummy tasks) has been scheduled. There are two main SGSs: serial SGS (task-increment procedure); and parallel SGS (time-increment procedure). Details of the general procedures for both SGS's and theoretical results can be found in Kolisch [1996a]. In this research the serial SGS, first introduced by Kelley [1963], is implemented. Parallel SGSs cannot be used when task splitting is allowed since in parallel SGSs, a schedule is only analyzed at the finish time of scheduled tasks. The time between task finish times may contain resources that a portion of a task to be scheduled.

The serial SGS consists of $g = 1,..,J$ stages. In each stage, one task and its associated mode is selected and scheduled at the earliest precedence and resource feasible time. Associated with each stage, two disjoint sets are calculated. The schedule set $S_g$ consists of the tasks that have already been scheduled. The decision set $D_g$ consists of the tasks that are eligible for scheduling. At each step $g$, let:

$J$ = the set of all tasks including two dummy tasks,

$M_j$ = the set of executable modes for task $j$,

$A(t)$ = the set of active tasks that are being processed at time $t$,

$S_g$ = the set of tasks that have already been scheduled,

$P_j$ = the set of predecessors of task $j$,

$D_g$ = the decision set that contains all eligible tasks, $D_g = \{ j \notin S_g \mid P_j \subseteq S_g \}$,

$R$ = the set of all renewable resources,

$K_r$ = the capacity of resource $r$,

$k_{j,m,r}$ = the resource demand for resource $r$ of task $j$, executed in mode $m$,

$\tilde{K}_r(t)$ = the remaining capacity of resource $r$ at time instant $t$, $\tilde{K}_r(t) = K_r - \sum_{j \in A(t)} k_{j,m,r}$,

$F_g$ = the set of all finish times, $F_g = \{FT_j \mid j \in S_g\}$, and

$T$ = the upper bound of the project completion time, $T = \sum_{j=2}^{J-1} \max_{m_j=1}^{M_j} \{d_{j,m}\}$

The serial SGS developed may be described with the following pseudo-code.

Initialization: $FT_1 = 0, S_1 = \{1\}$,
For $g = 2$ to $J-1$,

1. Calculate $D_g, F_g$, and update $\tilde{K}_r(t) \quad \forall r \in R, \forall t \in [1, \max\{F_g\}]$.
2. Select a task $j \in D_g$ and its associated mode $m_j \in M_j$ based on a priority rule and a selection method.
3. Calculate the earliest start time of task $j$,
   $EST_j = \min\{t \mid k_{j,m,r} \le \tilde{K}_r(t), \forall r \in R\}$.
4. Calculate the finish time of task $j$,
   $d = d_{j,m}$
   For $t = EST_j$ to $LFT_j$,
   If $k_{j,m,r} \le \tilde{K}_r(t), \forall r \in R$,
   $d = d - 1$,
   End
   If $d = 0$,
   $FT_j = t$
   Break
   End
   End
5. Update the schedule set $S_g = S_{g-1} \cup \{j\}$

End
Calculate the project makespan $FT_J = \max_{h \in P_J} \{FT_h\}$.

The initialization step schedules the dummy start task $j = 1$ with its completion time of 0, and puts it in the partial schedule. At each step $g$, the decision set $D_g$, the set of finish

times $F_g$, and the remaining capacities $\tilde{K}_r(t)$ for each resource $r$ at the finish times $t \in F_g$ are calculated. Then, a task $j$ with an associated mode $m_j$ is selected from the decision set using a priority rule and a selection method. The earliest start time of task $j$ is calculated by determining the earliest precedence and resource feasible time that the first unit subtask of the selected task $j$ may be scheduled. The finish time of task $j$ within $[EST_j, LFT_j]$ is then calculated as the earliest resource feasible time that the last subtask of task $j$ may be scheduled. $LFT_j$ denotes the latest finish time as calculated from a backward recursion from an upper bound of the project's finish time $T$. The serial SGS terminates at step $g = J$, when all tasks are scheduled.

There are pertinent lemmas in the literature about serial SGSs. Pinson et al. [1994] has proved that the serial SGS has polynomial time complexity. Kolisch [1996a] has shown that the serial SGS always generates feasible schedules and always generates *active* schedules, which are schedules where no task can be started earlier without delaying some other tasks. Sprecher [1995a] proved that a optimal schedule will always be in the set of *active* schedules.

### 5.1.2 Priority Rules

Priority rules are used to select between task-mode options competing for the allocation of scarce resources. A *priority rule* consists of a priority value and a priority function. Priority values for each task-mode option are calculated using functions, which are based on some numerical measures related to properties of the tasks, the level of resources, the completion of the project, or to task splitting. Selecting a particular task-mode option, once priority values have been calculated, is done with a priority function (usually the maximum or minimum priority value). In addition, since this heuristic is for multi-mode

problems, two distinct sets of priority values are used. One set for tasks, and the other set for modes.

5.1.2.1 Task Priority Rules and Mode Priority Rules

Several task priority rules have been found to perform well (Davis and Patterson [1975], Boctor [1993], Kolisch [1996a, 1996b]). Five task priority rules are used in the heuristic. Table 5.1 summarizes these rules. For modes, three priority rules are used (SFM is adapted from Boctor [1993], where LTRU and LCRU are developed in this research). Table 5.2 summarizes the definitions and classification of these mode priority rules.

Table 5.1 Task priority rules used

| Priority Function | Numerical Measure | Definition | Classification | |
|---|---|---|---|---|
| | | | Static vs. Dynamic | Local vs. Global |
| MIN | Latest Finish Time (LFT) | $LFT_j$ [1] | Static | Local |
| MIN | Latest Start Time (LST) | $LFT_j - \min_{m=1}^{M_j} d_{j,m}$ | Static | Global |
| MAX | Most Total Successor (MTS) | $\lvert\overline{S}_j\rvert$ | Static | Local |
| MAX | Greatest Rank Positional Weight (GRPW) | $\max_{m=1}^{M_j} d_{j,m} + \sum_{i \in S_j} \left( \max_{m=1}^{M_i} d_{i,m} \right)$ | Dynamic | Local |
| MIN | Slack (SLK) | $LST_j - EST_j$ | Static | Local |

[1] Latest finish time (LFT) calculation is given in Section 4.2.2.2.

Table 5.2 Mode priority rules used

| Priority Function | Numerical Measure | Definition | Classification | |
|---|---|---|---|---|
| | | | Static vs. Dynamic | Local vs. Global |
| MIN | Shortest Feasible Mode (SFM) | $\min_{m=1}^{M_j} d_{j,m}$ | Static | Local |
| MIN | Least Total Resource Usage (LTRU) | $\sum_{r=1}^{R}\left(k_{j,r,m}\times d_{j,m}\right)$ | Static | Local |
| MIN | Least Critical Resource Usage[1] (LCRU) | $k_{j,r^*,m}\times d_{j,m}$ [1] | Dynamic | Local |

[1] The determination of the critical resource $r^*$ will be explained in Section 5.1.2.3.

5.1.2.2 Combining Task and Mode Priority Values

For multi-mode projects, the priority values from task and mode priority rules must be combined into one priority value for each task mode option. Both task and mode priority values are normalized on a 0-1 scale before they are combined. Table 5.3 summarizes the normalization functions for the task and mode priority values.

Let $v(j)$ and $v'(j)$ denote task priority values and normalized task priority values for the tasks in the decision set, $j \in D_g$,

$v(m)$ and $v'(m)$ denote mode priority values and normalized mode priority values for the candidate mode, $m \in \{1,..,M_j\}$, for each eligible task $j$.

Table 5.3 Normalization for task and mode priority values

| | Priority function | |
|---|---|---|
| | Minimum (MIN) | Maximum (MAX) |
| Task | $v'(j) = \frac{(1/v(j))}{\sum_{j \in D_g} (1/v(j))}$ | $v'(j) = \frac{v(j)}{\sum_{j \in D_g} v(j)}$ |
| Mode | $v'(m) = \frac{(1/v(m))}{\sum_{m=1}^{M_j} (1/v(m))}$ | $v'(m) = \frac{v(m)}{\sum_{m=1}^{M_j} v(m)}$ |

Using the normalization functions given in Table 5.3, the higher normalized priority value always gets higher priority. From Table 5.1 and Table 5.2, five task priority values and three mode priority values are combined to make 15 task-mode combination priority values. Given a task priority rule and – a mode priority rule, a task-mode priority value can be calculated and combined as outlined next.

Step 1: For each task $j \in D_g$, determine the mode with the highest normalized mode priority value (with respect to the selected mode priority rule).

Step 2: For tasks $j \in D_g$, normalize the mode priority values across tasks with respect to the mode values selected in Step 1.

Step 3: For each task $j \in D_g$, calculate the normalized *task* priority value with respect to the selected task priority rule.

Step 4: Combine the task priority value and mode priority value,

$$v(j,m) = v'(j) + v'(m) \tag{5.1}$$

**Example 5.1** (continued from Example 4.1) Suppose the task priority rule selected is MIN-LFT and the mode priority rule selected is MIN-SPT. In the first step of the priority rule-based heuristic, $D_g = \{2,3\}$.

Step 1: For task 2, the selected mode is mode 1, because mode 2 is inefficient by bounding rule 1. From Table 4.1, task 3 executed in mode 1 takes 9 time units, and 10 time units in mode 2, hence, the normalized mode priority value for mode 1 and mode 2 are $\frac{(1/9)}{(1/9)+(1/10)} = 0.5263$ and $\frac{(1/10)}{(1/9)+(1/10)} = 0.4737$, respectively. Therefore, the selected mode for task 3 is mode 1.

Step 2: For task 2 and task 3, the normalized mode priority values are $\frac{(1/8)}{(1/8)+(1/9)} = 0.5294$ and $\frac{(1/9)}{(1/8)+(1/9)} = 0.4706$, respectively (task 2 executed in mode 1 takes 8 unit durations).

Step 3: For task 2 and task 3, from Figure 4.5 (a) the LFT are 54 and 45, respectively. Hence, the normalized task priority values are $\frac{(1/54)}{(1/54)+(1/45)} = 0.4545$ for task 2 and $\frac{(1/45)}{(1/54)+(1/45)} = 0.5455$ for task 3.

Step 4: Compare task 2 executed in mode 1 with task 3 executed in mode 1, the task-mode priority values are $0.5294+0.4545 = 0.9839$ and $0.4706+0.5455 = 1.0161$, respectively. Therefore task 3, executed in mode 1, is selected to be scheduled in this step of the serial SGS.

### 5.1.3 Moving Resource Strength

An objective of this heuristic is to incorporate task splitting, but to only do so when task splitting is likely to be of benefit. The results from the computational experiments indicate that task splitting has a significant chance of improving a schedule when resources are tight. A new concept, called *moving resource strength* ( *MRS* ), is developed for measuring the tightness of resources within a moving window as a schedule is generated with a serial SGS. The MRS is used to determine if a task should be split and scheduled at an earlier time, or to wait until the task can be completed without splitting.

Moving resource strength ( *MRS* ) is similar to the ProGen parameter called resource strength ( *RS* ). *MRS* is a dynamic measure that quantifies the resource situation at every step of project scheduling. Consider a scheduling step in the priority rule-based heuristic.

Let $t_1$ denote the earliest time that any eligible task in the decision set, $D_g$, can be started, $t_1 = \min_{j \in D_g} EST_j$

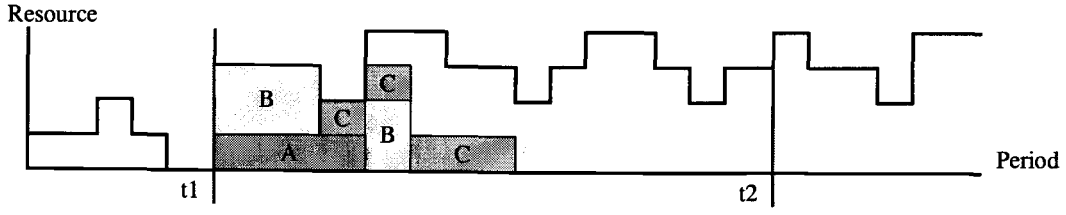
$t_2$ denote the earliest time that any eligible task in the decision set, $D_g$, can be finished, $t_1 = \min_{j \in D_g} LFT_j$

Moving resource strength or resource $r$ is calculated as,

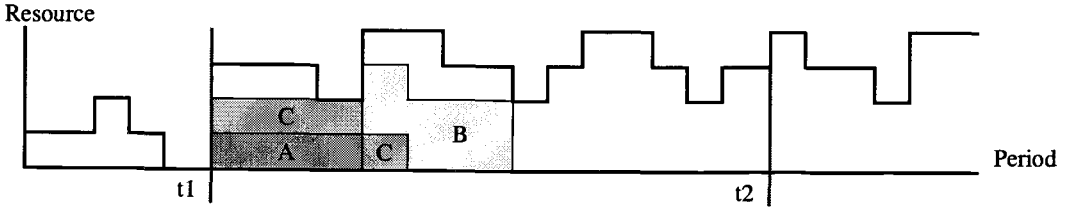
$$MRS_r = \frac{\bar{K}_r}{K_r^{\max}} \tag{5.2}$$

where $\bar{K}_r$ is the average resource capacity for resource $r$ over a time window $t_1$ and $t_2$, $\bar{K}_r = \frac{\sum_{t=t_1}^{t_2} K_{r,t}}{t_2 - t_1}$; and $K_r^{\max}$ is the maximum resource capacity for resource $r$ over a time window $t_1$ and $t_2$, $K_r^{\max} = \max_{t=t_1}^{t_2} K_{r,t}$.

Similar to $RS$ (equation 4.3), $MRS$ is a normalized on a 0-1 scale and measures the availability of each resource with respect to the maximum available capacity. When the $MRS$ of any resource is close to zero, it implies that that particular resource is tight, while a $MRS$ close to one implies that the resource under consideration is sufficiently available.

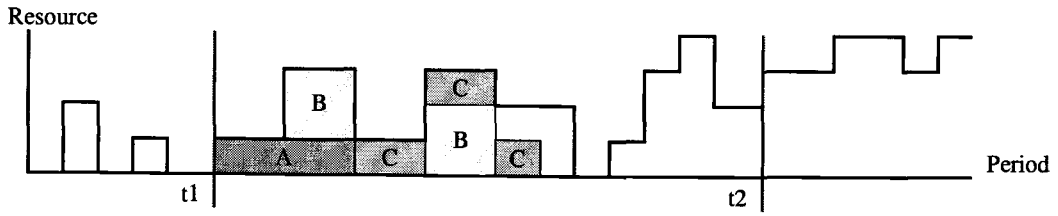



Within time window ( t1, t2 ), MRS = 0.75
Schedule: A - B - C, resulting in B being split
Makespan = t1 + 8

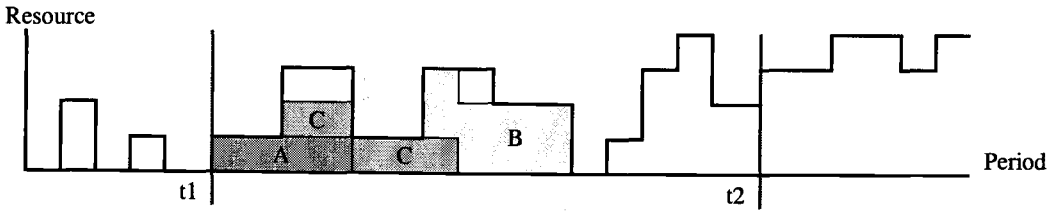



Within time window ( t1, t2 ), MRS = 0.75
Schedule: A - C - B, no task is split.
Makespan = t1 + 8

Figure 5.1 Resource situations where task splitting provides no benefit.




Within time window ( t1, t2 ), MRS = 0.50
Schedule: A - B - C; task B is split.
Makespan = t1 + 9




Within time window ( t1, t2 ), MRS = 0.50
Schedule: A - C - B, no task is split.
Makespan = t1 + 10

Figure 5.2 Resource situations where task splitting improves the schedule.

The concept of *MRS* can be illustrated using Figure 5.1 and Figure 5.2. In Figure 5.1 the *MRS* of the resource within the time window $[t_1, t_2]$ is 0.75. When starting with the same resource profile the upper schedule (A-B-C) with task B split yields the same makespan as the lower schedule (A-C-B) where no task is split. The *MRS* of the resource in Figure 5.2 is 0.50, which indicates a tighter resource situation within the time window $[t_1, t_2]$. The upper schedule (A-B-C) with task B being split reduces the makespan by one time unit over the lower schedule (A-C-B) with no task split. Task splitting is beneficial under this tighter resource situation.

Controlling the number of task splits in the heuristic is an important feature. Although task splitting is assumed to have no penalty, this is seldom true in reality. *MRS* is implemented in the priority rule-based heuristic by adding a task splitting priority value, $v(ts)$, to the task-mode priority value, $v(j,m)$, obtained in (5.1),

$$v'(j,m) = v(j,m) + v(ts) \tag{5.3}$$

When using the *MRS* in the $v(ts)$ function, the most *critical resource* associated with each task-mode option is identified. For a task-mode option, the critical resource is the resource with the lowest value of *MRS*, given that the demand for that resource exists, $k_{j,m,r} > 0$. This definition of the critical resource is the same as the critical resource in the mode priority rule LCRU in Table 5.2.

Let $MRS_{\min}$ denote the moving resource strength of the task-mode option under consideration, $MRS_{\min} = \min_{r \in R}\{MRS_r \mid k_{j,m,r} > 0\}$.

$split_{j,m}$ denote a binary variable for each task-mode option that takes on a value of one if selecting that task-mode option results in the selected task being split when executed in the selected mode; and takes on a value of zero, otherwise.

The task splitting priority function can be defined as,

$$v(ts) = \begin{cases} (1 - MRS_{\min}) \times split_{j,m} & ,0 < MRS_{\min} \le \delta_B \\ 0 & ,\delta_B < MRS_{\min} \le \delta_P \\ -MRS_{\min} \times split_{j,m} & ,\delta_P < MRS_{\min} \le 1 \end{cases} \tag{5.4}$$

where $\delta_B$ and $\delta_P$ denote the benefit/penalty region for $v(ts)$.

With the use of $\delta_B$ and $\delta_P$, $v(ts)$ will return a positive value proportional to $(1-MRS_{\min})$ when $MRS$ is lower than $\delta_B$, implying that the task-mode option under consideration will get more priority value for task being split under a tight resource situation (MRS is low). On the other hand, $v(ts)$ will give a negative value proportional to $-MRS_{\min}$ when $MRS$ is higher than $\delta_P$, indicating that the task-mode option will get less priority value if the task is split under a plentiful resource situation ($MRS$ is high).

An alternative form of $v(ts)$ can be constructed by setting $\delta_B$ and $\delta_P$ as one parameter,

$$v(ts) = \begin{cases} (1-MRS_{\min}) \times split_{j,m} & ,0 < MRS_{\min} \le \delta \\ -MRS_{\min} \times split_{j,m} & ,\delta < MRS_{\min} \le 1 \end{cases} \tag{5.5}$$

These benefit/penalty parameters ($\delta_B$ and $\delta_P$, or $\delta$) must be adjusted empirically (Section 5.2.1). The value of $v(ts)$ from (5.4) or (5.5), may make $v'(j,m)$ negative. In that case, $v'(j,m)$ will be replaced with a small constant $\varepsilon > 0$ to assure a non-zero priority value for that task-mode option. A non-zero priority value is only necessary in a stochastic priority rule-based method (presented in Section 5.1.4) to assure a non-zero selection probability for that task-mode option.

**Example 5.1** Continuing the previous example, the task-mode priority value for task 2 executed in mode 1 is 0.9839, and for task 3 executed in mode 1 is 1.0161. From the resource profile shown in Figure 4.2, and the resource demands in Table 4.3, executing task 2 in mode 1 will result in task 2 being split ($split_{2,1} = 1$), while executing task 3 in mode 1 will result in task 3 not being split ($split_{3,1} = 0$).

From the resource profile and resource demand, task 2 executed in mode 1 only requires resource 2, hence, $MRS_2$ is used as $MRS_{\min}$ in the $v(ts)$ function. The time window for task 2 and task 3 is from $t_1 = 0$ to $t_2 = \min\{LFT_2, LFT_3\} = \min\{54, 45\} = 45$, which results in $MRS_2 = 0.7714$ (calculated from the complete resource profile not shown in Figure 4.2).

Suppose $\delta_P$ is set to be 0.67, then task 2 executed in mode 1 will get the task splitting priority value, $v(ts) = -0.7714$, resulting in the total task-mode priority value of $0.9839 - 0.7714 = 0.2125$. The total task-mode priority value for task 3 executed in mode 1 stays the same. Therefore, task 3 executed in mode 1 is selected as the task-mode option to be scheduled in this step.

### 5.1.4 Selection Methods

The last component of the priority rule-based heuristic is how to select a task-mode option from the decision set based on task-mode priority values. Two selection methods are available:

1) Deterministic - selects the task-mode option with the highest priority value.
2) Stochastic - selects a task-mode option according to some sampling methods, where the selection probability for each task-mode option is computed based on its priority value.

Three commonly used sampling schemes are:

1) Random sampling (RS) - each candidate in the decision set has the same selection probability,
2) Biased random sampling (BRS) - a tasks selection probability is its normalized priority value (relative to the priority values of other task that may be scheduled),

3) Regret based random sampling (RBRS) - uses a tasks priority value to calculate its probability of selection with a "regret function".

Drexl et al. [1991] proposed a modification to the regret function by adding two parameters to the regret function. One parameter is a small constant that is added directly to the regret function to ensure non-zero probability of selection for each task. The other parameter controls the selection probabilities that the regret function will cause no bias when the parameter takes on a high value and will yield maximum bias with a zero value of the parameter. Further details of all the selection methods can be found in Drexl [1991], Kolisch [1995, and others], and Schirmer and Riesenberg [1997].

### 5.1.5 Constructing Priority Rule-Based Heuristic

This section presents descriptions of three versions of a priority-rule based heuristic that permits task splitting and implements the moving resource strength concept.

#### 5.1.4.1 Deterministic Multi-Pass Priority Rule-based Heuristic

The simplest deterministic priority rule-based method is to use only one combination of task priority rule and mode priority rule and select the task-mode option with the highest priority value. This method is called a "single-pass" method. The disadvantage of the single pass method is that the priority value is calculated from only one combination of rules, which may not work well in some project situations. Thus, a better deterministic method is the multi-pass method that uses multiple combinations of task priority rules and mode priority rules.

A deterministic multi-pass priority rule method executes a serial SGS several times. Each time a different task priority rule/mode priority rule combination is used. All five task priority rules and three mode priority rules (Table 5.1 and Table 5.2) are combined to make fifteen task-mode priority rule combinations as listed in Table 5.4. The motivation for this method is from the observation that no single priority rule generates good schedules for every project. By employing several task-mode priority rule combinations, a better schedule may be obtained. Examples of research where this idea has been used are Boctor [1990, 1993], Ulusoy and Ozdamar [1989], Thomas and Salhi [1997].

Table 5.4 Task-mode priority rule combinations

| Task priority rule:Mode priority rule Combinations | | |
|---|---|---|
| MIN-LFT: MIN-SFM | MIN-LFT: MIN-LTRU | MIN-LFT:MIN-LCRU |
| MIN-LST: MIN-SFM | MIN-LST: MIN-LTRU | MIN-LST: MIN-LCRU |
| MAX-MTS: MIN-SFM | MAX-MTS: MIN-LTRU | MAX-MTS: MIN-LCRU |
| MAX-GRPW: MIN-SFM | MAX-GRPW: MIN-LTRU | MAX-GRPW: MIN-LCRU |
| MIN-SLK: MIN-SFM | MIN-SLK: MIN-LTRU | MIN-SLK: MIN-LCRU |

The deterministic multi-pass priority rule-based heuristic can be described with the following psuedo-code .

Initialization: $F_{best} = \varnothing, F_J = \varnothing$
For *rule* = 1 to 15,

I. Perform serial SGS with *rule*
Reset: $FT_1 = 0, S_1 = \{1\}$,
For $g = 2$ to $J-1$,

1. Calculate $D_g, F_g$, and update $\tilde{K}_r(t)$ $\quad \forall r \in R, \forall t \in [1, \max\{F_g\}]$.
2. Calculate $v'(j,m) = v(j,m) + v(ts)$ $\quad \forall j \in D_g, \forall m_j \in \{1,..,M_j\}$.

3. Select a task-mode option, $j \in D_g$, $m_j \in M_j$, with the highest $v'(j,m)$, (Ties are broken arbitrarily).
4. Calculate earliest start time of task $j$ executed in mode $m$, $EST_j = \min\{t \mid k_{j,m,r} \le \tilde{K}_r(t), \forall r \in R\}$
5. Calculate finish time of task $j$ executed in mode $m$, $d = d_{j,m}$

For $t = EST_j$ to $LFT_j$, {

If $k_{j,m,r} \le \tilde{K}_r(t), \forall r \in R$,
$d = d - 1$,
End
If $d = 0$,
$FT_j = t$
Break
End
}
End

6. Update the schedule set, $S_g = S_{g-1} \cup \{j\}$.

End

II. Calculate the project makespan obtained using *rule*, $FT_J = \max_{h \in P_J}\{FT_h\}$

End
Calculate the best overall project makespan among all *rule*, $F_{best} = \min\{FT_J\}$

where $F_{best}$ denotes the best makespan among the 15 task-mode priority rules.

5.1.4.2 Stochastic Multi-Pass Priority Rule-Based Heuristic with Biased Random Sampling

With a deterministic method, running the method multiple times with the same combination of rules will generate the same schedule. Using random sampling in the task-mode option selection process generates different schedules. For each task-mode priority rule combination, the serial SGS is repeated for a certain number of iterations. In an iteration a task-mode option is selected according to the following probability:

$$p(j,m)=\frac{v'(j,m)}{\sum_{i=1}^{|D_g|}v'(i,m)} \tag{5.6}$$

The heuristic can be described with the following pseudo-code.

Initialization 1: $F_{best}=\varnothing, F_J=\varnothing$
For $rule$ = 1 to $n_{rule}$,

I. Initialization 2: $F_{best,rule}=\varnothing$
For $iter$ = 1: $n_{iter}$,

i) Perform serial SGS with *rule*
Reset: $FT_1=0, S_1=\{1\}$,
For $g=2$ to $J-1$,

1. Calculate $D_g, F_g$, and update $\tilde{K}_r(t)$ $\quad \forall r\in R, \forall t\in[1,\max\{F_g\}]$.
2. Calculate $v'(j,m)=v(j,m)+v(ts)$ $\quad \forall j\in D_g, \forall m_j\in\{1,..,M_j\}$.
3. Calculate $p(j,m)=\dfrac{v'(j,m)}{\sum_{i=1}^{|D_g|}v'(i,m)}$ $\quad \forall j\in D_g, \forall m_j\in\{1,..,M_j\}$.
4. Select a task-mode option, $j\in D_g$, $m_j\in M_j$, with the probability of selection, $p(j,m)$.
5. Calculate earliest start time of task $j$ executed in mode $m$, $EST_j=\min\{t \mid k_{j,m,r}\le\tilde{K}_r(t), \forall r\in R\}$
6. Calculate finish time of task $j$ executed in mode $m$, $d=d_{j,m}$

For $t=EST_j$ to $LFT_j$, {
If $k_{j,m,r}\le\tilde{K}_r(t), \forall r\in R$,
$d=d-1$,
End
If $d=0$,
$FT_j=t$
Break
End

}

End

7. Update the schedule set, $S_g = S_{g-1} \cup \{j\}$.

End

ii) Finish one iteration, calculate the project makespan, $FT_J = \max_{h \in P_J}\{FT_h\}$.

End

II. Finish one *rule*, calculate the best makespan from $n_{iter}$ iterations,

$$F_{best,rule} = \min\left\{F_{best,rule}, \max_{h \in P_J}(F_h)\right\}.$$

End

Calculate the best overall makespan among all *rule*, $F_{best} = \min_{rule=1}^{n_{rule}}\left(F_{best,rule}\right)$

where $F_{best,rule}$ and $F_{best}$ denotes the best makespan out of $n_{iter}$ trials for a *rule*, and the best overall makespan, respectively.

The parameter $n_{rule}$ is the number of task-mode priority rule combinations to be used in the heuristic. In section 5.2.2, an experiment is conducted to determine the best set of task-mode priority rule combinations out of the possible fifteen used. Another parameter that must be considered for the stochastic priority rule-based heuristic is the number of iterations, $n_{iter}$, that the serial SGS will be repeated for each particular task-mode rule. In the study conducted by Schirmer and Riesenberg [1997], several versions of biased random sampling and regret-based random sampling are tested with both serial and parallel SGSs. The numbers of iterations were varied from 10 to 100. It was found that for a set of good priority rules the performance of the serial SGS becomes dominant (over a parallel SGS) after the number of iteration exceeds 70. In the heuristic developed in this research, the number of iterations was set at 100. Although, increasing the number of iterations in a stochastic serial SGS will continue to improve the solution quality, but the rate of improvement decreases rapidly with the price of increased computational time.

As described in Section 5.1.3 it is possible that the task-mode priority value $v'(j,m)$ may be negative. In that case $v'(j,m)$ will be replaced with $\varepsilon = 0.05$ to assure positive probability of selection for all task-mode priority options. Giving a bad task-mode priority value a small probability of selection will not significantly affect the overall performance of the heuristic. Solving a set of 50 randomly selected small projects using $\varepsilon = 0.01$ and 0.05, gave results that were statistically the same.

5.1.4.3 Stochastic Multi-Pass Priority Rule-Based Heuristics with Regret-Based Random Sampling

The sampling scheme implemented in this heuristic is similar to biased random sampling, except instead of assigning the probability for each task-mode option proportional to the task-mode priority values, the probability of selection for each task mode is calculated from the regret function, $r(j,m)$. This is a function of the task-mode priority values of the eligible tasks.

$$r(j,m) = p(j,m) - \min_{i \in D_g} p(j,m) + \frac{1}{\beta} \times \min_{i \in D_g} p(j,m) \tag{5.7}$$

$$p'(j,m) = \frac{r(j,m)}{\sum_{i=1}^{|D_g|} r(i,m)} \tag{5.8}$$

$p(j,m)$ is calculated as in (5.6), which are the probabilities used in biased random sampling. $p'(j,m)$ is the probability used in regret-based random sampling
The heuristic's description is the same as in Section 5.1.4.2, except $p(j,m)$ is replaced by $r(j,m)$, when selecting a task-mode option in the serial SGS. The parameter $\beta$ is the fraction of the minimum of $v'(j,m)$ that is added in order to assure that the selection

probability for each task-mode option in the decision set is greater than zero. Good values for this parameter are indicated from test in Section 5.2.3.

## 5.2 TESTING THE HEURISTIC ON SMALL PROJECTS

This experiment is conducted to test the performance of the heuristic methods as well and to empirically adjust some algorithm parameters. The small projects from experiment 2 (Chapter 4) are used in this experiment. The objectives of the experiment are 1) to tune the benefit/penalty region used in the task splitting priority function, 2) to determine whether or not the use of moving resource strength ( $MRS$ ) in the task splitting priority function ( $v(ts)$ ) can control the number of task splits while maintaining the solution quality, 3) to determine what combinations of task-mode priority rules perform best with respect to the project makespan, and 4) to test the performance of the three versions of the heuristic developed.

### 5.2.1 Adjusting the Benefit/Penalty Parameters

The task splitting priority function $v(ts)$ has two forms defined in (5.4) and (5.5). In (5.4) the function is divided into three regions by the parameters $\delta_B$ and $\delta_P$. In (5.5) the function has two regions separated by the parameter $\delta$. Table 5.5 shows the values of these parameters tested.

Table 5.5 Tested values for the parameters in $v(ts)$

| $v(ts)$ | Parameter | Tested Value | | | | | |
|---|---|---|---|---|---|---|---|
| 3-Region | $\delta_B$ | 0.25 | 0.25 | 0.25 | 0.33 | 0.33 | 0.33 |
| | $\delta_P$ | 0.50 | 0.67 | 0.75 | 0.50 | 0.67 | 0.75 |
| 2-Region | $\delta$ | 0.25 | 0.33 | 0.50 | | | |

In Table 5.5, there are nine values for the parameter of $v(ts)$, six combinations of $\delta_B$ and $\delta_P$ for the 3-region function and three values of $\delta$ for the 2-region function. All nine sets of tested parameters are implemented in the deterministic multi-pass priority rule-based heuristics (DM) with fifteen task-mode priority rule combinations. In addition, the DM is implemented *without* $v(ts)$ so that there is no control over the number of task splitting. This is to verify if the $v(ts)$ function works as intended.

The responses are project makespans, $ms$, and the number of task splits, $n_{split}$. The number of task splits for each project is a count of the number of times the tasks included in a project are split (one task can be split more than once). Multiple comparison analysis is used to compare the responses.

The objective of the comparisons is to find the best sets of benefit/penalty parameters that will result in a minimum number of task splits while preserving the solution quality (makespan). Three multiple comparisons are performed using the project makespan as a response variable on three data sets, including the whole data set containing 3240 observations, the "hard" data sets containing 1080 observations (where $RS$ is 0.25 or 0.50 and the $RF$ is 0.75 or 1.0), and the "split-only" data set containing the 2239 observations where task splitting occurs (i.e. the problem instances, which no task are split, are excluded).

Tukey's method for all pairwise comparisons is used to compare 10 sets of different values of benefit/penalty parameters as shown in Table 5.5. The results from multiple comparisons on all three data sets are the same. There is no statistical difference in the means of the project makespan among the 10 set of benefit/penalty parameters, p-value = 0.05. This implies that there is no statistical difference in the performance of the deterministic multi-pass priority rule-based heuristics with any of the parameters tested.

The groupings of these parameters are therefore done according to the average number of task splits. Tukey's method is used for multiple comparisons on the number of task splits. The results are shown in Table 5.6.

Table 5.6 Multiple comparisons using the Tukey method

| $\delta_B / \delta_P$, or $\delta$ | All instances | | Hard instances | | Split-only instances | |
|---|---|---|---|---|---|---|
| | Grouping | Avg. $n_{split}$ | Grouping | Avg. $n_{split}$ | Grouping | Avg. $n_{split}$ |
| 0.33 | x | 1.5852 | x | 2.7176 | x | 2.2939 |
| 0.25/0.5 | x | 1.5880 | x | 2.7232 | x | 2.2979 |
| 0.25 | x | 1.5886 | x | 2.7176 | x | 2.2988 |
| 0.33/0.5 | x | 1.5920 | x | 2.7232 | x | 2.3037 |
| 0.5 | x | 1.5944 | x | 2.7222 | x | 2.3073 |
| 0.25/0.67 | x | 1.6679 | x x | 2.8306 | x | 2.4136 |
| 0.33/0.67 | x | 1.6679 | x x | 2.8269 | x | 2.4136 |
| 0.25/0.75 | x | 1.8438 | x x | 3.1565 | x | 2.6682 |
| 0.33/0.75 | x | 1.9602 | x x | 3.1565 | x | 2.6717 |
| no $v(ts)$ | x | 1.9602 | x | 3.3296 | x | 2.8365 |

Three multiple comparisons are also performed using a non-parametric method, permutation test. The results, shown in Table 5.7, provide groups of means by the number of task splits (level of significance of 0.05). The results from (a) and (b) are almost identical, and the results from (c) are slightly different. The benefit/penalty parameter that

is selected for the priority rule-based heuristic is the 2-region parameter $\delta = 0.33$. This is because, 1) $\delta = 0.33$ belongs to the group with the minimum number of task splits in all three comparisons using both methods, and within this group $\delta = 0.33$ ranks first in all three comparisons, 2) $\delta = 0.33$ also performs well relative to makespan (ranks 2nd in all comparisons, even though there is no statistical differences in project makespan among the 10 groups), and 3) using this parameter will results in a 2-region $v(ts)$ function, which is simpler to implement than the 3-region function.

The results indicate that the DMP without $v(ts)$ ranks last, and is statistically significantly different from most of the parameters tested. Therefore, the ability to control the number of task splits using $v(ts)$ is verified.

Table 5.7 Multiple comparison using the permutation test

| $\delta_B / \delta_P$, or $\delta$ | Grouping | $n_{split}$ rank | Avg. $n_{split}$ | $ms$ rank | Avg. $ms$ |
|---|---|---|---|---|---|
| 0.33 | x | 1 | 1.5852 | $2^{tie}$ | 25.6475 |
| 0.25/0.5 | x | 2 | 1.5880 | 1 | 25.6454 |
| 0.25 | x | 3 | 1.5886 | 8 | 25.6574 |
| 0.33/0.5 | x | 4 | 1.5920 | 6 | 25.6525 |
| 0.5 | x | 5 | 1.5944 | $2^{tie}$ | 25.6475 |
| 0.25/0.67 | x | 6 | 1.6679 | 9 | 25.7028 |
| 0.33/0.67 | x | 7 | 1.6679 | 10 | 25.7191 |
| 0.25/0.75 | x | 8 | 1.8438 | $2^{tie}$ | 25.6475 |
| 0.33/0.75 | x | 9 | 1.8463 | 5 | 25.6481 |
| no $v(ts)$ | x | 10 | 1.9602 | 7 | 25.6528 |

(a) Multiple comparisons on all projects

| $\delta_B / \delta_P$, or $\delta$ | Grouping | $n_{split}$ rank | Avg. $n_{split}$ | *ms* rank | Avg. *ms* |
|---|---|---|---|---|---|
| 0.33 | x | 1 | 2.2939 | $2^{tie}$ | 25.64753 |
| 0.25/0.5 | x | 2 | 2.2979 | 1 | 25.64537 |
| 0.25 | x | 3 | 2.2988 | 8 | 25.65741 |
| 0.33/0.5 | x | 4 | 2.3037 | 6 | 25.65247 |
| 0.5 | x | 5 | 2.3073 | $2^{tie}$ | 25.64753 |
| 0.33/0.67 | x | 6 | 2.4136 | 10 | 25.71914 |
| 0.25/0.67 | x | 7 | 2.4136 | 9 | 25.70278 |
| 0.25/0.75 | x | 8 | 2.6682 | $2^{tie}$ | 25.64753 |
| 0.33/0.75 | x | 9 | 2.6717 | 5 | 25.64815 |
| no $v(ts)$ | x | 10 | 2.8365 | 7 | 25.65278 |

(b) Multiple comparisons on "hard" projects

| $\delta_B / \delta_P$, or $\delta$ | Grouping | $n_{split}$ rank | Avg. $n_{split}$ | *ms* rank | Avg. *ms* |
|---|---|---|---|---|---|
| 0.25 | x | $1^{tie}$ | 2.7176 | 8 | 25.65741 |
| 0.33 | x | $1^{tie}$ | 2.7176 | $2^{tie}$ | 25.64753 |
| 0.50 | x | 3 | 2.7222 | $2^{tie}$ | 25.64753 |
| 0.25/0.50 | x | 4 | 2.7231 | 1 | 25.64537 |
| 0.33/0.50 | x | 5 | 2.7231 | 6 | 25.65247 |
| 0.33/0.67 | x x | 6 | 2.8269 | 10 | 25.71914 |
| 0.25/0.67 | x x x | 7 | 2.8306 | 9 | 25.70278 |
| 0.33/0.75 | x x x | 8 | 3.1528 | 5 | 25.64815 |
| 0.25/0.75 | x x | 9 | 3.1565 | $2^{tie}$ | 25.64753 |
| no $v(ts)$ | x | 10 | 3.3296 | 7 | 25.65278 |

(c) Multiple comparisons on "split-only" projects

### 5.2.2 Determining the Best Set of Task-Mode Priority Rules

The objective of this multiple comparison is to select the best set of task-mode priority rules to be used in the stochastic priority rule-based methods. Tukey's method is applied to project makespans obtained from performing the deterministic single-pass priority rule-based method multiple times, with each single-pass using one of the fifteen task-

mode priority rules. Table 5.8 summarizes the results from multiple comparisons on the makespans obtained from each single pass (grouping is done with p-value of 0.05).

The results from the multiple comparison shows that the first 7 rules in Table 5.8 perform best and are statistically the same with respect to project makespan. The permutation test is also performed (detail results are not shown). The results indicate that the first 4 rules are the best set of task-mode priority rules. To be conservative, 7 rules are used in the stochastic methods.

Table 5.8 Multiple comparisons on the project makespan of 15 single-passes

| Task-Mode Priority Rule | Grouping | | | | Avg. *ms* | *ms* rank |
|---|---|---|---|---|---|---|
| MAX-GRPW: MIN-SFM | x | | | | 27.7352 | 1 |
| MIN-LFT: MIN-SFM | x | | | | 27.7389 | 2 |
| MAX-MTS: MIN-LCRU | x | | | | 27.7670 | 3 |
| MIN-SLK: MIN-LTRU | x | x | | | 28.2547 | 4 |
| MIN-LST: MIN-SFM | x | x | | | 28.3574 | 5 |
| MIN-SLK: MIN-SFM | x | x | | | 28.3583 | 6 |
| MAX-GRPW: MIN-LCRU | x | x | | | 28.3920 | 7 |
| MIN-LST: MIN-LTRU | | x | | | 28.5620 | 8 |
| MIN-LFT:MIN-LCRU | | x | | | 28.6296 | 9 |
| MAX-MTS: MIN-LTRU | | x | | | 28.8719 | 10 |
| MAX-MTS: MIN-SFM | | | x | | 31.6241 | 11 |
| MIN-LFT: MIN-LTRU | | | x | | 31.6361 | 12 |
| MIN-SLK: MIN-LCRU | | | x | | 31.8994 | 13 |
| MIN-LST: MIN-LCRU | | | | x | 32.5216 | 14 |
| MAX-GRPW: MIN-LTRU | | | | x | 32.8148 | 15 |

### 5.2.3 Testing the Performance of the Heuristic

The three versions of the heuristic developed that are tested in this section are the following.

1. Deterministic multi-pass priority rule-based heuristic (DM) with 15 task-mode rules
2. Biased random sampling priority rule-based heuristic (BRS) with 7 task-mode rules
3. Regret-based random sampling priority rule-based heuristic (RBRS) with 7 task-mode rules

One issue that must be addressed before conducting this test involves the parameter $\beta$ of the regret function in the RBRS. From the regret function (5.7), $r(j,m)$ is calculated from $p(j,m)$, which is the probability of the biased random sampling. Thus, $r(j,m)$ is calculated from normalized values (because $p(j,m)$'s add up to one). The value of parameter $\beta$ (which indicates the magnitude of bias on the probability of selection) that will be considered are 2, 3, and 4. This is because the values of $\beta$ greater than 4 will not bias the probability of selection much further. Therefore, this experiment will test the RBRS with $\beta$ of 2, 3, and 4.

Table 5.9 Summary of results for the tested heuristics

| Criteria | DM | BRS | RBRS/2 | RBRS/3 | RBRS/4 |
|---|---|---|---|---|---|
| Optimal found | 33.09% | 74.57% | 73.36% | 72.38% | 72.56% |
| Avg.off.opt | 2.2630 | 0.4549 | 0.4802 | 0.5080 | 0.5139 |
| Avg.off.opt.% | 7.5846% | 1.5938% | 1.6816% | 1.7652% | 1.7737% |
| Max.off.opt | 40 | 12 | 12 | 12 | 17 |
| Max.off.opt.% | 44.44% | 25.00% | 25.00% | 25.00% | 25.00% |

(a) Results on all projects

| Criteria | DM | BRS | RBRS/2 | RBRS/3 | RBRS/4 |
|---|---|---|---|---|---|
| Optimal found | 12.04% | 58.80% | 56.76% | 54.17% | 54.54% |
| Avg.off.opt | 3.8065 | 0.8231 | 0.8750 | 0.9509 | 0.9769 |
| Avg.off.opt.% | 11.1018% | 2.5486% | 2.6916% | 2.9145% | 2.9775% |
| Max.off.opt | 40 | 12 | 12 | 12 | 17 |
| Max.off.opt.% | 43.48% | 25.00% | 25.00% | 25.00% | 25.00% |

(b) Results on "hard" projects

Multiple comparisons (Tukey's method) are performed twice, one with all problem instances and the other with only hard instances. It is found that the results from both multiple comparisons are similar. As expected, the performance (project makespan) of all stochastic methods are statistically superior to that of the deterministic method, p-value < 0.05. However, the performance of all stochastic methods are not statistically different from one another, p-value = 0.05. Table 5.9 provides a summary of results.

## 5.3 TESTING THE HEURISTIC METHODS ON MEDIUM AND LARGE PROJECTS

### 5.3.1 Generating Medium and Large Projects

The modified ProGen is used to generate the projects. Table 5.10 shows ProGen parameters used. There are three problem sets generated, a 30-task set, a 60-task set, and a 90-task set. Network complexity is also used for generating these problem sets. For these problem sets, instead of holding the network complexity, resource factors, and resource strength at fixed levels, these parameters are randomly selected from a uniform distribution between the minimum level and maximum level of each parameter given in

Table 5.10. For these problem sets the levels of resource factor and resource strength are set such that "hard" problems are generated (resource constraints are tight). Tight resource situations are where task splitting can benefit the schedule.

Table 5.10 Modified ProGen for generating medium and large projects.

| Base data | Minimum | Maximum |
|---|---|---|
| Number of tasks | 30, 60, 90 | 30, 60, 90 |
| Modes per task | 4 | 4 |
| Task duration | 1 | 10 |
| Number of resources | 4 | 4 |
| Number of resource requested | 1 | 4 |
| Number of task 1 (dummy start) successors | 2 | 5 |
| Number of task $j$ successors | 1 | 5 |
| Number of predecessors of task $J$ (dummy finish) | 2 | 5 |
| Number of predecessors of task $j$ | 1 | 5 |
| Network complexity, $NC$ | 1.5 | 2.0 |
| Resource factor, $RF$ | 0.5 | 1.0 |
| Resource strength, $RS$ | 0.25 | 0.5 |
| Percent resource vacations, $V_P$ | 10% | 30% |
| Percent short resource vacation, $V_L$ | 0 | 100% |
| Short vacation length | 1 | 2 |
| Long vacation length | 3 | 5 |
| Number of problem instances | 100 | 100 |

### 5.3.2 Testing the Heuristics on Medium and Large Projects

The three heuristics tested in the previous experiment are tested on the medium and large projects in this experiment. These priority rule-based heuristics include the deterministic multi-pass method (DM), biased random sampling method (BRS), and regret-based random sampling method (RBRS). Because the RBRS with $\beta$ of 2, 3, and 4 do not have

statistical significant difference in their performance, only RBRS with $\beta$ of 2 is tested here. Also, all methods use the $v(ts)$ function with a 2-region $\delta$ parameter of 0.33.

All three methods are tested without allowing task splitting, with task splitting but without the $v(ts)$ function, and with task splitting and the $v(ts)$ function. The three responses of interest, include the project makespans ($ms$), the number of task splits ($n_{split}$), and the computational time ($cpu_{time}$).

Tables 5.11, 5.12, and 5.13 summarize the results of multiple comparisons on project makespans (Tukey's method and the permutation test both performed with a level of significant of 0.05), the number of task splits (permutation test), and computation time, respectively.

Table 5.11 Summary of results on the project makespans

| Methods | Grouping | | | | Avg *ms* | SD *ms* |
|---|---|---|---|---|---|---|
| BRS/no_$v(ts)$ | x | | | | 41.65 | 9.1700 |
| RBRS/no_$v(ts)$ | x | | | | 41.67 | 9.1221 |
| RBRS | x | x | | | 42.91 | 11.1120 |
| BRS | x | x | | | 43.21 | 11.1467 |
| BRS/no_ts | x | x | x | | 45.97 | 12.4993 |
| RBRS/no_ts | x | x | x | | 46.19 | 12.5470 |
| DM/no_$v(ts)$ | | x | x | | 47.23 | 10.8496 |
| DM | | | x | x | 50.51 | 13.0937 |
| DM/no_ts | | | | x | 55.81 | 15.6248 |

(a) Number of tasks = 30

| Methods | Grouping | | | Avg *ms* | SD *ms* |
|---|---|---|---|---|---|
| BRS/no_$v(ts)$ | x | | | 71.46 | 21.7244 |
| RBRS/no_$v(ts)$ | x | | | 71.49 | 21.7081 |
| BRS | x | x | | 77.65 | 27.7818 |
| RBRS | x | x | | 78.34 | 27.9589 |
| DM/no_$v(ts)$ | x | x | | 81.56 | 23.8164 |
| BRS/no_ts | x | x | | 82.22 | 31.9379 |
| RBRS/no_ts | x | x | x | 83.91 | 37.7485 |
| DM | | x | x | 90.72 | 32.5964 |
| DM/no_ts | | | x | 101.28 | 50.2535 |

(b) Number of tasks = 60

| Methods | Grouping | | | Avg *ms* | SD *ms* |
|---|---|---|---|---|---|
| BRS/no_$v(ts)$ | x | | | 93.43 | 26.6793 |
| RBRS/no_$v(ts)$ | x | | | 93.78 | 26.3007 |
| BRS | x | x | | 102.65 | 33.8151 |
| RBRS | x | x | | 102.65 | 34.0482 |
| BRS/no_ts | x | x | | 104.54 | 30.5738 |
| RBRS/no_ts | x | x | | 105.48 | 30.3560 |
| DM/no_$v(ts)$ | | x | | 106.29 | 29.3885 |
| DM | | x | x | 117.93 | 36.4775 |
| DM/no_ts | | | x | 128.21 | 36.2269 |

(c) Number of tasks = 90

Table 5.12 Summary of results on the number of task splits

| 30-task set | | | 60-task set | | | 90-task set | | |
|---|---|---|---|---|---|---|---|---|
| Method | Avg | SD. | Method | Avg. | SD. | Method | Avg. | SD. |
| RBRS | 4.47 | 2.91 | DM | 12.32 | 6.95 | DM | 19.25 | 7.48 |
| BRS | 4.54 | 2.46 | BRS | 13.00 | 6.42 | RBRS | 20.36 | 6.72 |
| DM | 4.63 | 3.00 | RBRS | 13.78 | 7.68 | BRS | 21.20 | 7.25 |
| RBRS/ no_$v(ts)$ | 7.71 | 3.97 | DM/ no_$v(ts)$ | 19.37 | 7.63 | DM/ no_$v(ts)$ | 31.13 | 9.37 |
| DM/ no_$v(ts)$ | 7.91 | 3.62 | RBRS/no_ $v(ts)$ | 21.43 | 8.16 | RBRS/ no_$v(ts)$ | 34.40 | 10.86 |
| BRS/ no_$v(ts)$ | 7.97 | 3.98 | BRS/no_ $v(ts)$ | 22.01 | 9.03 | BRS/ no_$v(ts)$ | 34.46 | 10.13 |

Table 5.13 Summary of results on the computational time

| Method | Iteration | 30-task set | | | 60-task set | | | 90-task set | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Avg. | SD. | Max | Avg. | SD. | Max | Avg. | SD. | Max |
| DM/ no_ts | 15 | 1.33 | 0.05 | 1.52 | 4.02 | 0.19 | 4.95 | 8.09 | 0.34 | 9.30 |
| DM/ no_$v(ts)$ | 15 | 1.25 | 0.04 | 1.34 | 3.77 | 0.09 | 3.97 | 7.67 | 0.21 | 8.20 |
| DM | 15 | 3.12 | 0.33 | 3.94 | 11.08 | 1.76 | 17.52 | 26 | 4.76 | 41 |
| BRS/ no_ts | 700 | 63 | 2.34 | 71 | 190 | 8.46 | 225 | 379 | 14 | 424 |
| BRS/ no_$v(ts)$ | 700 | 58 | 1.57 | 61 | 178 | 4.04 | 188 | 363 | 9 | 394 |
| BRS | 700 | 156 | 16.0 | 203 | 568 | 81.5 | 813 | 1252 | 213 | 2003 |
| RBRS/ no_ts | 700 | 63 | 2.38 | 71 | 189 | 8.36 | 225 | 382 | 14 | 430 |
| RBRS/ no_$v(ts)$ | 700 | 59 | 2.47 | 71 | 175 | 3.83 | 184 | 361 | 9 | 390 |
| RBRS | 700 | 154 | 15.5 | 196 | 568 | 85 | 842 | 1263 | 224 | 2074 |

5.3.3 Conclusion from Testing the Heuristics on Medium and Large Projects

The results from Table 5.11, 5.12, and 5.13 can be interpreted as follows. In all three task sets,

- BRS without the $v(ts)$ function and RBRS without the $v(ts)$ function perform best and are statistical different from all three deterministic methods, p-value = 0.05. An exception is in 60-task set where DM without the $v(ts)$ function generates a better average project makespan than the two stochastic methods where task splitting is not allowed (statistically in the same group).
- All six stochastic methods cannot be statistically distinguished from one another, even in the cases where tasks are not allowed to split. However, if the multiple comparisons are done within only the six stochastic methods, the first two methods (BRS/no_$v(ts)$ and RBRS/no_$v(ts)$ ) statistically outperform RBRS/no_ts. The results are shown in Table 5.14.

Table 5.14 Multiple comparison on project makespans within the six stochastic methods.

| Method | 30-task set | | 60-task set | | 90-task set | |
|---|---|---|---|---|---|---|
| | Grouping | | Grouping | | Grouping | |
| BRS/no_$v(ts)$ | x | | x | | x | |
| RBRS/no_$v(ts)$ | x | | x | | x | |
| BRS | x | x | x | x | x | x |
| RBRS | x | x | x | x | x | x |
| BRS/no_ts | x | x | x | x | x | x |
| RBRS/no_ts | | x | | x | | x |

- The numbers of task splits for all methods with the $v(ts)$ function are statistically less than those of the methods without $v(ts)$ function (p-value = 0.05).

- Deterministic methods require less computation time due to a smaller number of iterations. Among stochastic methods, the ones with $v(ts)$ functions require more than three times as much computation time as the ones without $v(ts)$ functions.

Even though the performance of the stochastic methods are not statistically different when the $v(ts)$ function is used, when it is not used, and even when task splitting is not allowed, differences can be recognized and can be of practical value. When applying the stochastic methods to a real world problem, there is a tradeoff between controlling the number of splits and the quality of the solution obtained. There is also a trade off between the number of iterations and the quality of the solution. In a real situation when there are only a few scheduling problems to solve at a time, it is always a good strategy to let the algorithm run longer than what has been done in this experiment to obtain better solutions. In the next chapter, a real project will be scheduled to demonstrating the use of the developed heuristic.

### 5.3.4 Heuristic Schedules Vs. Human Developed Schedules

This last section presents another evaluation of the heuristics performance. In general, evaluation of a heuristic can be performed by comparing the heuristic solution with a known optimal solution if it exists, or by comparing the heuristic against lower bounds for the optimal solution. The optimal solutions for the medium to large projects are computationally infeasible due to problem complexity. Here, an evaluation of the heuristic on the medium sized projects is done by comparing the quality of the solution obtained by the heuristics against schedules developed by humans using hand calculations.

Ten people from different fields of study/expertise volunteered for this evaluation. A randomly selected project from the 30-task problem set in experiment 4 is used with task splitting allowed. Table 5.15 shows the results obtained from the evaluation.

From the results, the effectiveness and efficiency of the heuristic methods are obvious. All four stochastic methods outperform the humans, while deterministic methods perform at the same level as the humans. A significant benefit of using the heuristic methods is computation time. Heuristic methods generated the results in less than 3 minutes, whereas human computational times range from 2.5 hours to approximately 4 hours.

Table 5.15 Comparison of the solutions obtained by scheduling with heuristics and scheduling with human

| Method | Project makespan | Number of task splitting | Computational time |
|---|---|---|---|
| RBRS | 26 | 0 | 141.36 sec. |
| BRS | 26 | 0 | 145.5 sec. |
| DM | 33 | 4 | 1.844 sec. |
| RBRS/no_$v(ts)$ | 26 | 3 | 56.109 sec. |
| BRS/no_$v(ts)$ | 28 | 5 | 55.156 |
| DM/no_$v(ts)$ | 31 | 5 | 1.219 sec. |
| Person 1 | 31 | 4 | > 2.5 hours |
| Person 2 | 35 | 3 | > 2.5 hours |
| Person 3 | 33 | 3 | > 2.5 hours |
| Person 4 | 31 | 5 | > 2.5 hours |
| Person 5 | 29 | 6 | > 2.5 hours |
| Person 6 | 34 | 3 | > 2.5 hours |
| Person 7 | 35 | 2 | > 2.5 hours |
| Person 8 | 31 | 4 | > 2.5 hours |
| Person 9 | 30 | 4 | > 2.5 hours |
| Person 10 | 35 | 3 | > 2.5 hours |

# 6. A CASE STUDY

This chapter presents an application of the developed heuristic to an engineering design project within the General Motors (GM) Truck Engineering organization. The objective is to demonstrate the use of the developed heuristic on a real world problem. Details of the design project are introduced in Section 6.1 and the resulting schedule is given in Section 6.2.

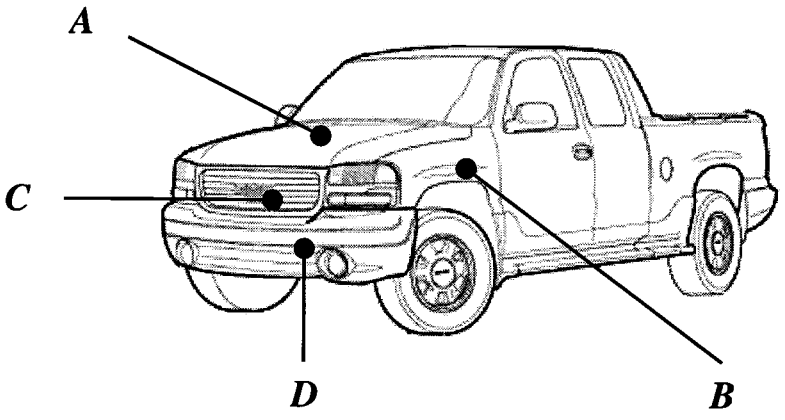
## 6.1 PROJECT DESCRIPTION AND DATA

The project[1] presented here is part of an engineering work order for the enhancement of General Motors' full-size pickup trucks. The enhancement involves modifications to the exterior front-end design of the vehicles as shown in Figure 6.1. The components involved include the Hood component (A), the Fender component (B), the Grille component (C), and the Front Bumper component (D). Figure 6.2 and Figure 6.3 lists the design tasks required for each area and their precedence relations as organized in Microsoft® Project. Each of the components is decomposed into parts and assembly elements, which are further broken down into parts. Design work is then assigned to each individual part.

For this project, parts required in the Hood area enhancement include the Hood Latch, Hood Outer part, and Hood Inner part. The enhancement in the Fender area includes the Fender Outer part and the Fender Extension part. For the Front Bumper component, the three parts to be modified are the Bar, Bracket, and Brace parts. The design work on

[1] Case study data is provided by Chih-Cheng Hsu (Senior Project Engineer)

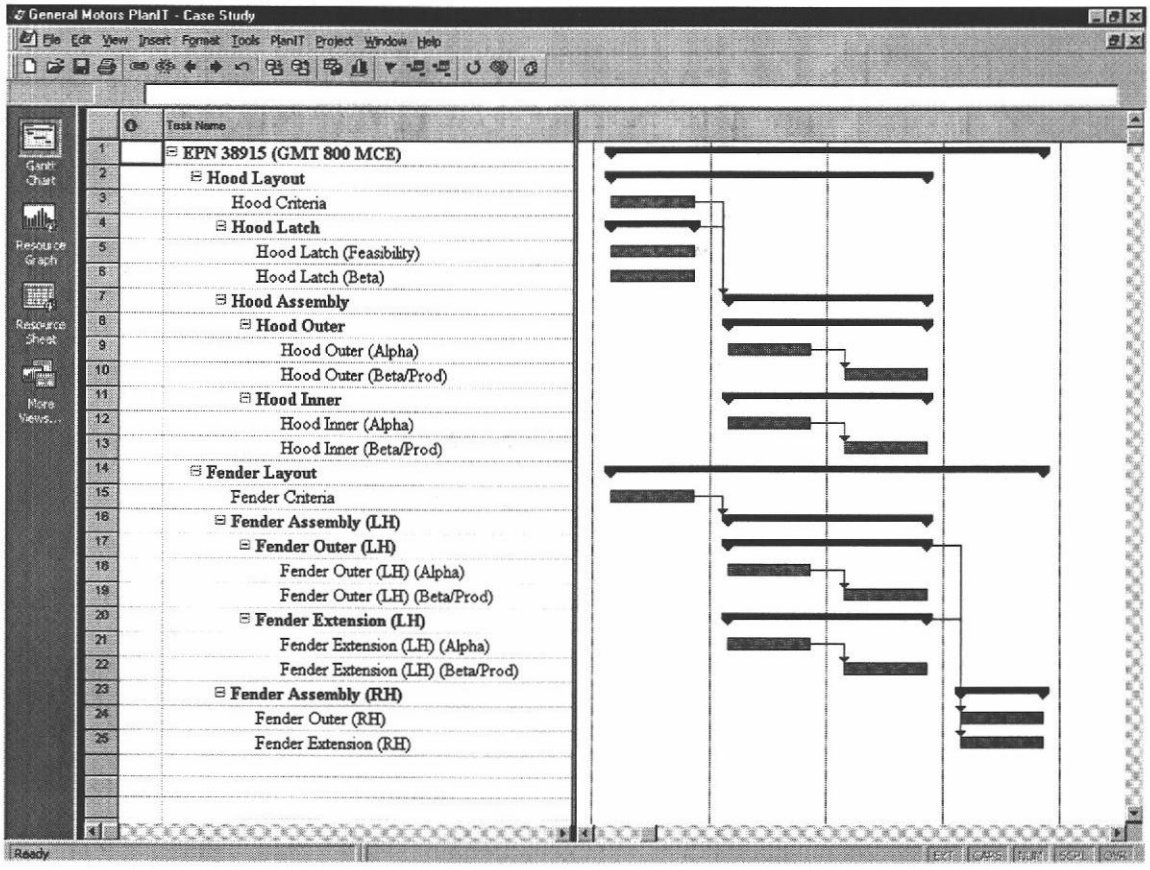
Figure 6.1 The components involved in the project




these parts can be categorized by different phases in the vehicle development (criteria study, alpha, and beta), or geographical locations (right-hand side and left-hand side). Parts are essential design units and each part has a unique control number. Design work on parts can be further categorized by the required design expertise, either senior designers (top skills) or regular designers (average skills), due to differences in the nature of the design.

In the project described, all tasks can be executed by either an average skilled, or a top skilled design engineer. In general, a higher skilled engineer completes a task faster than an average skilled engineer does. Possible task-mode options to execute each task are summarized in Table 6.1.

Figure 6.2 Tasks required for the Hood and Fender components




In addition, the design engineers participating in this project are also involved in other projects or scheduled training. Thus, they will not be available at all periods for this design project. The unavailable (vacation) schedule for a calendar year is shown in Figure 6.4.

Management requires project completion within 45 weeks. If that is infeasible, management will consider contracting new design engineers (possibly up to 5 additional senior engineers and 5 additional regular engineers) for the project. The objective is to schedule all design tasks to finish the project before its deadline, or determine whether additional engineers must be hired.

Figure 6.3 Task required for the Grilled and Front Bumper components

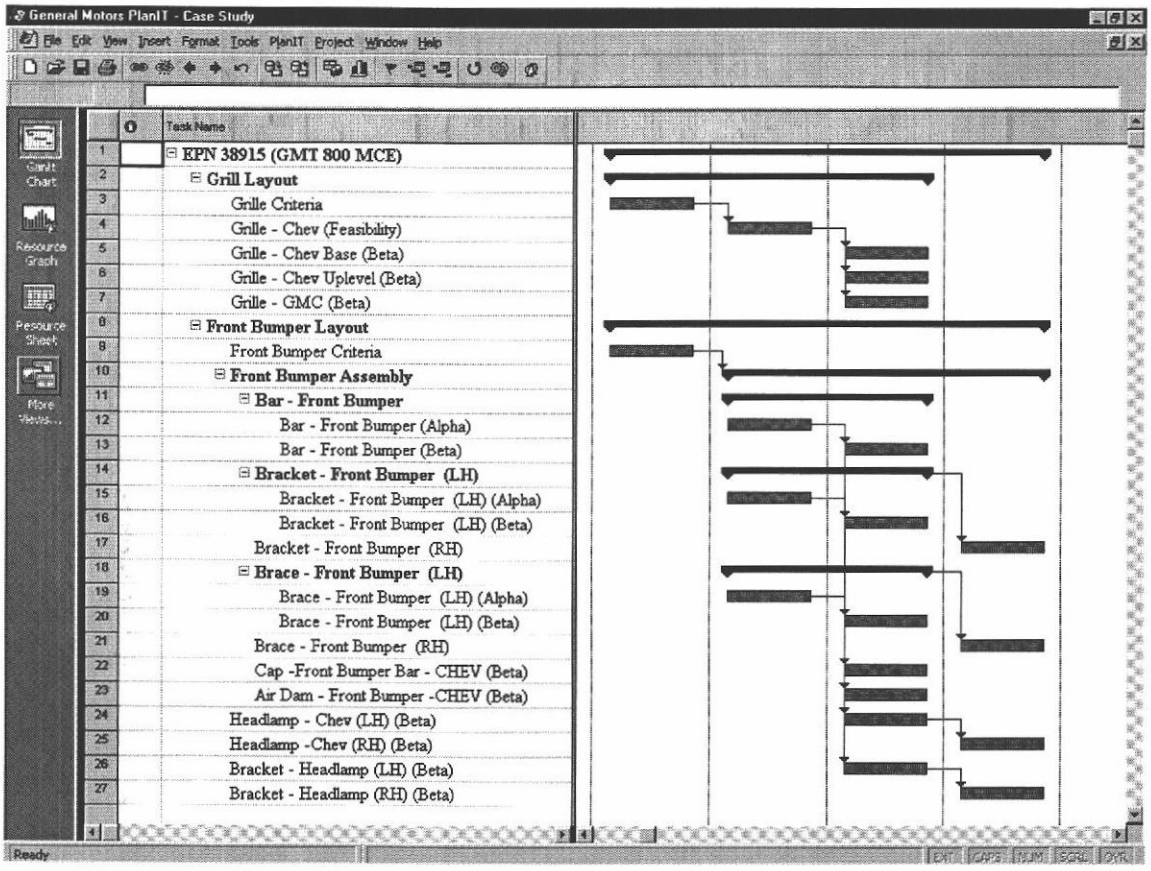


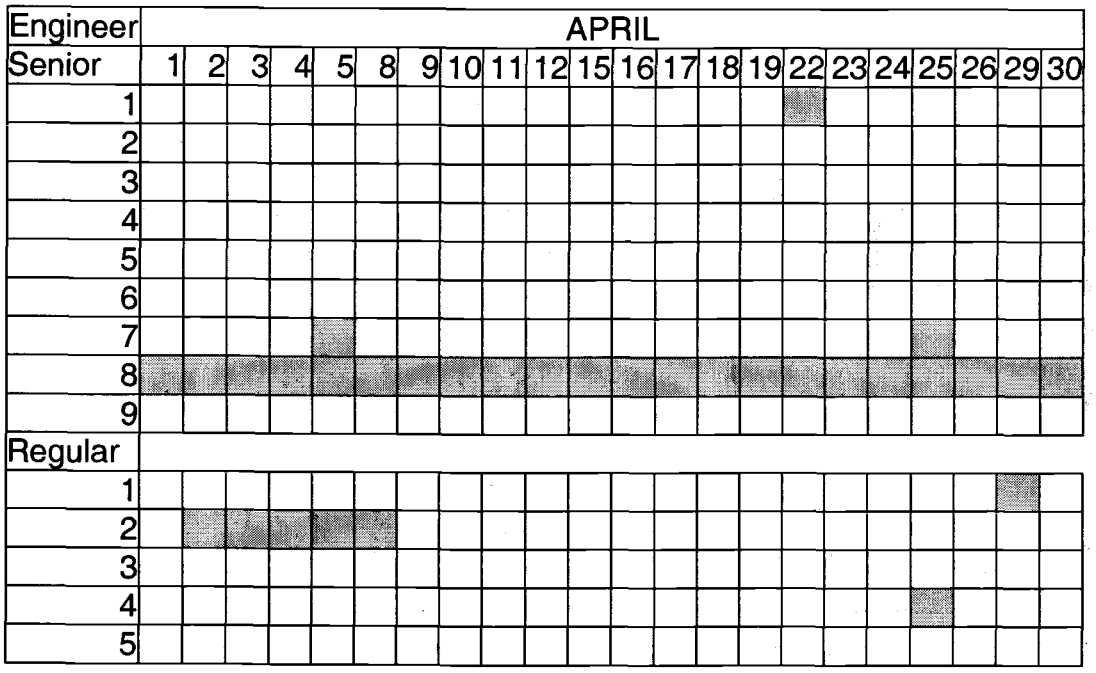

Figure 6.4 Human resource vacations schedule

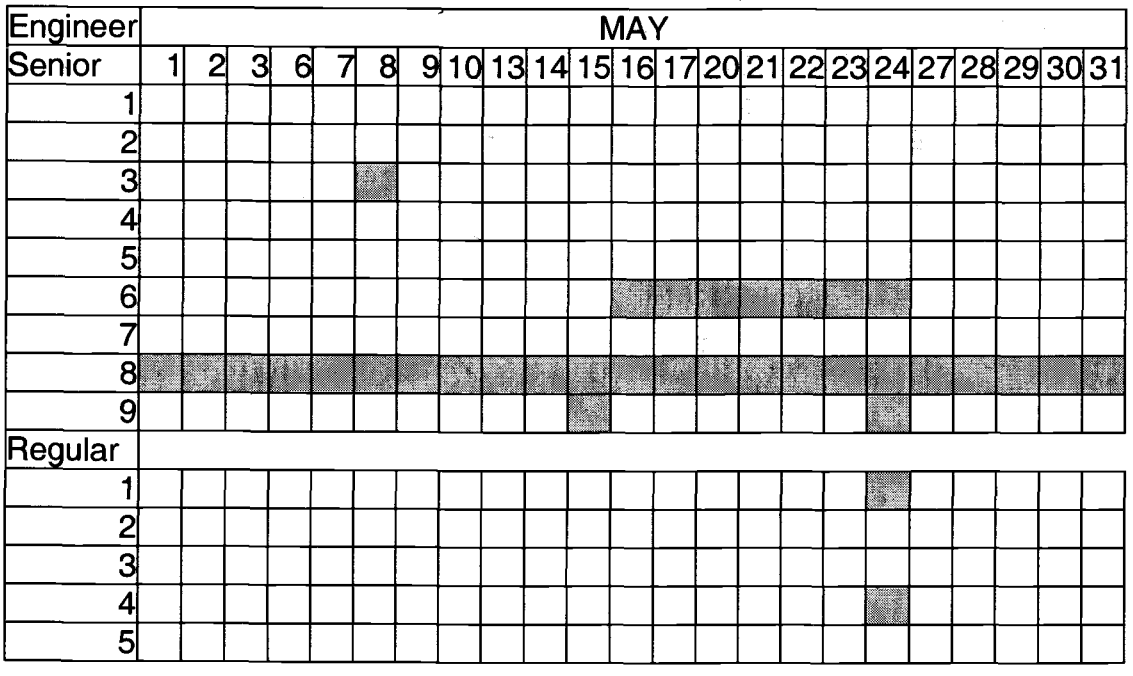
| Engineer | JANUARY | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Senior | 1 | 2 | 3 | 4 | 7 | 8 | 9 | 10 | 11 | 14 | 15 | 16 | 17 | 18 | 21 | 22 | 23 | 24 | 25 | 28 | 29 | 30 | 31 |
| 1 | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | | | | | | | | |

| Regular | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | | | | | | |

| Engineer | FEBRUARY | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Senior | 1 | 4 | 5 | 6 | 7 | 8 | 11 | 12 | 13 | 14 | 15 | 18 | 19 | 20 | 21 | 22 | 25 | 26 | 27 | 28 |
| 1 | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | | | | | |
| Regular | | | | | | | | | | | | | | | | | | | | |
| 1 | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | | | |

| Engineer | MARCH | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Senior | 1 | 4 | 5 | 6 | 7 | 8 | 11 | 12 | 13 | 14 | 15 | 18 | 19 | 20 | 21 | 22 | 25 | 26 | 27 | 28 | 29 |
| 1 | | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | | | | | | |
| Regular | | | | | | | | | | | | | | | | | | | | | |
| 1 | | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | | | | |


Engineer
APRIL
Senior
1 2 3 4 5 8 9 10 11 12 15 16 17 18 19 22 23 24 25 26 29 30
1
2
3
4
5
6
7
8
9
Regular
1
2
3
4
5


Engineer
MAY
Senior
1 2 3 6 7 8 9 10 13 14 15 16 17 20 21 22 23 24 27 28 29 30 31
1
2
3
4
5
6
7
8
9
Regular
1
2
3
4
5

| Engineer | JUNE | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Senior | 3 | 4 | 5 | 6 | 7 | 10 | 11 | 12 | 13 | 14 | 17 | 18 | 19 | 20 | 21 | 24 | 25 | 26 | 27 | 28 |
| 1 | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | | | | | |
| Regular | | | | | | | | | | | | | | | | | | | | |
| 1 | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | | | |

| Engineer | JULY | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Senior | 1 | 2 | 3 | 4 | 5 | 8 | 9 | 10 | 11 | 12 | 15 | 16 | 17 | 18 | 19 | 22 | 23 | 24 | 25 | 26 | 29 | 30 | 31 |
| 1 | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | | | | | | | | |
| Regular | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | | | | | | |

| Engineer | AUGUST | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Senior | 1 | 2 | 5 | 6 | 7 | 8 | 9 | 12 | 13 | 14 | 15 | 16 | 19 | 20 | 21 | 22 | 23 | 26 | 27 | 28 | 29 | 30 |
| 1 | | | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | | | | | | | |
| Regular | | | | | | | | | | | | | | | | | | | | | | |
| 1 | | | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | | | | | |

| Engineer | SEPTEMBER | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Senior | 2 | 3 | 4 | 5 | 6 | 9 | 10 | 11 | 12 | 13 | 16 | 17 | 18 | 19 | 20 | 23 | 24 | 25 | 26 | 27 | 30 |
| 1 | | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | | | | | | |
| Regular | | | | | | | | | | | | | | | | | | | | | |
| 1 | | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | | | | |

| Engineer | OCTOBER | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Senior | 1 | 2 | 3 | 4 | 7 | 8 | 9 | 10 | 11 | 14 | 15 | 16 | 17 | 18 | 21 | 22 | 23 | 24 | 25 | 28 | 29 | 30 | 31 |
| 1 | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | | | | | | | | |
| Regular | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | | | | | | |

| Engineer | NOVEMBER | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Senior | 1 | 4 | 5 | 6 | 7 | 8 | 11 | 12 | 13 | 14 | 15 | 18 | 19 | 20 | 21 | 22 | 25 | 26 | 27 | 28 | 29 |
| 1 | | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | | | | | | |
| Regular | | | | | | | | | | | | | | | | | | | | | |
| 1 | | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | | | | |

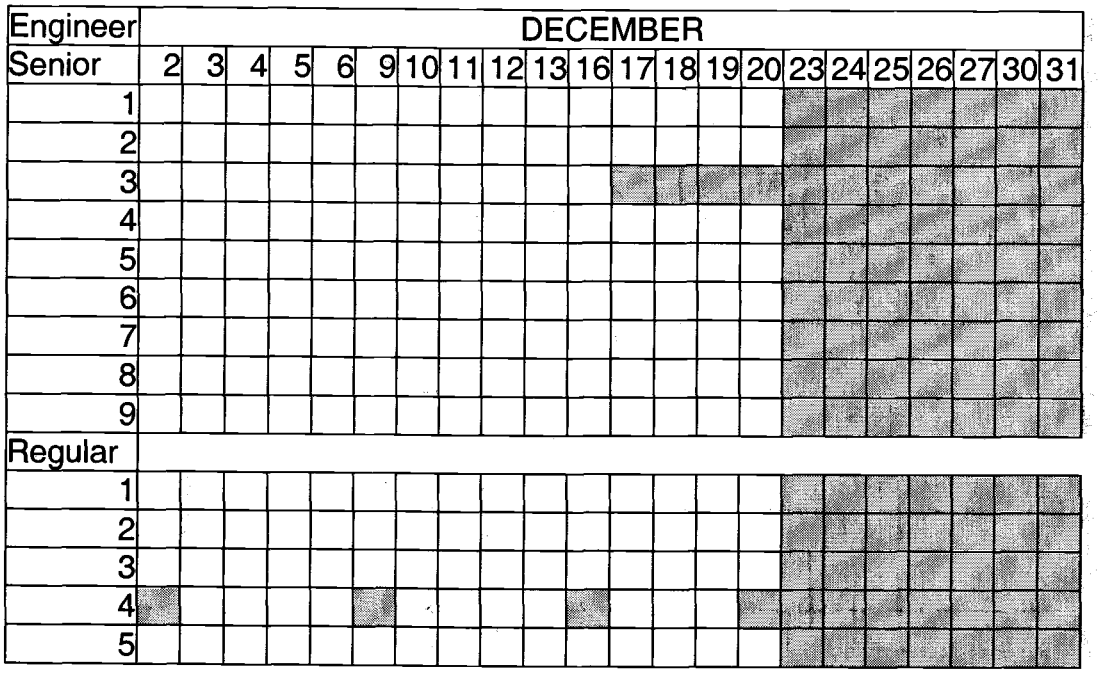



Table 6.1 Duration and human resource requirement for each task-mode options

| Task name | Mode | Duration (weeks) | Resource requirement | |
|---|---|---|---|---|
| | | | Regular engineer | Senior engineer |
| 1. Hood Criteria | 1 | 4 | 0 | 1 |
| | 2 | 6 | 1 | 0 |
| 2. Hood Latch (Feasibility) | 1 | 4 | 0 | 1 |
| | 2 | 6 | 1 | 0 |
| 3. Hood Latch (Beta) | 1 | 4 | 0 | 1 |
| | 2 | 6 | 1 | 0 |
| 4. Hood Outer (Alpha) | 1 | 16 | 0 | 1 |
| | 2 | 20 | 1 | 0 |
| 5. Hood Outer (Beta/Prod) | 1 | 16 | 0 | 1 |
| | 2 | 20 | 1 | 0 |
| 6. Hood Inner (Alpha) | 1 | 16 | 0 | 1 |
| | 2 | 20 | 1 | 0 |
| 7. Hood Inner (Beta/Prod) | 1 | 16 | 0 | 1 |
| | 2 | 20 | 1 | 0 |
| 8. Fender Criteria | 1 | 4 | 0 | 1 |
| | 2 | 6 | 1 | 0 |

| Task name | Mode | Duration (weeks) | Resource requirement | |
|---|---|---|---|---|
| | | | Regular engineer | Senior engineer |
| 9. Fender Outer (LH) (Alpha) | 1 | 14 | 0 | 1 |
| | 2 | 17 | 1 | 0 |
| 10. Fender Outer (LH) (Beta/Prod) | 1 | 14 | 0 | 1 |
| | 2 | 17 | 1 | 0 |
| 11. Fender Extension (LH) (Alpha) | 1 | 10 | 0 | 1 |
| | 2 | 13 | 1 | 0 |
| 12. Fender Extension (LH) (Beta/Prod) | 1 | 10 | 0 | 1 |
| | 2 | 13 | 1 | 0 |
| 13. Fender Outer (RH) | 1 | 1 | 0 | 1 |
| | 2 | 1 | 1 | 0 |
| 14. Fender Extension (RH) | 1 | 1 | 0 | 1 |
| | 2 | 1 | 1 | 0 |
| 15. Grille Criteria | 1 | 4 | 0 | 1 |
| | 2 | 6 | 1 | 0 |
| 16. Grille – Chev (Feasibility) | 1 | 12 | 0 | 1 |
| | 2 | 15 | 1 | 0 |
| 17. Grille – Chev Base (Beta) | 1 | 16 | 0 | 1 |
| | 2 | 20 | 1 | 0 |
| 18. Grille – Chev Uplevel (Beta) | 1 | 14 | 0 | 1 |
| | 2 | 17 | 1 | 0 |
| 19. Grille – GMC (Beta) | 1 | 12 | 0 | 1 |
| | 2 | 15 | 1 | 0 |
| 20. Front Bumper Criteria | 1 | 5 | 0 | 1 |
| | 2 | 6 | 1 | 0 |
| 21. Bar – Front Bumper (Alpha) | 1 | 16 | 0 | 1 |
| | 2 | 20 | 1 | 0 |
| 22. Bar – Front Bumper (Beta) | 1 | 14 | 0 | 1 |
| | 2 | 17 | 1 | 0 |
| 23. Bracket – Front Bumper (LH) (Alpha) | 1 | 6 | 0 | 1 |
| | 2 | 8 | 1 | 0 |
| 24. Bracket – Front Bumper (LH) (Beta) | 1 | 6 | 0 | 1 |
| | 2 | 8 | 1 | 0 |
| 25. Bracket – Front Bumper (RH) | 1 | 1 | 0 | 1 |
| | 2 | 1 | 1 | 0 |
| 26. Brace – Front Bumper (LH) (Alpha) | 1 | 6 | 0 | 1 |
| | 2 | 8 | 1 | 0 |
| 27. Brace – Front Bumper (LH) (Beta) | 1 | 6 | 0 | 1 |
| | 2 | 8 | 1 | 0 |

| Task name | Mode | Duration (weeks) | Resource requirement | |
|---|---|---|---|---|
| | | | Regular engineer | Senior engineer |
| 28. Brace – Front Bumper (RH) | 1 | 1 | 0 | 1 |
| | 2 | 1 | 1 | 0 |
| 29. Cap – Front Bumper Bar – Chev (Beta) | 1 | 14 | 0 | 1 |
| | 2 | 17 | 1 | 0 |
| 30. Air Dam – Front Bumper – Chev (Beta) | 1 | 11 | 0 | 1 |
| | 2 | 14 | 1 | 0 |
| 31. Headlamp – Chev (LH) (Beta) | 1 | 7 | 0 | 1 |
| | 2 | 10 | 1 | 0 |
| 32. Headlamp – Chev (RH) (Beta) | 1 | 1 | 0 | 1 |
| | 2 | 1 | 1 | 0 |
| 33. Bracket – Headlamp (LH) (Beta) | 1 | 14 | 0 | 1 |
| | 2 | 17 | 1 | 0 |
| 34. Bracket – Headlamp (RH) (Beta) | 1 | 1 | 0 | 1 |
| | 2 | 1 | 1 | 0 |

## 6.2 COMPUTATIONAL RESULTS

The project given has the following properties.

1. Network complexity: The total number of arcs in the network is 70 and the total number of nodes is 36. Therefore, network complexity of the project is $\frac{70}{36} = 1.94$.
2. Resource factor: this parameter takes on a value of 0.5, because each mode only uses either one of the two skill levels of engineers.
3. Resource strength: $K_r, K_r^{\min}$, and $K_r^{\max}$ are calculated to be $K_1 = 5, K_2 = 9, K_1^{\min} = K_2^{\min} = 0, K_1^{\max} = K_2^{\max} = 14$, from Table 6.1, Equation (4.4), and Equation (4.5), respectively. This yields resource strengths of $RS_1 = \frac{5-0}{14-0} = 0.36$ and $RS_2 = \frac{9-0}{14-0} = 0.64$.

4. Percent resource vacations: The total number of workdays for this calendar year is $262 \times 14 = 3668$, and the total number of vacation days within this year is 455, which results in a percent resource vacation of $\frac{455}{3668} \times 100\% = 12.40\%$.
5. Resource vacation length: All 455 vacation days can be summarized as given in Table 6.2. It can be seen that 60 out of 99 vacations are ranging from 1 to 3 days where the rest of the vacation lengths are more than 3 days. Thus, the percent of short resource vacations is estimated to be $\frac{60}{99} \times 100\% = 60.61\%$.

From these parameters, a schedule that allows task splitting has a good chance of improving the project makespan because most of the vacations are short and one of the resources are tight. The project is solved with all nine methods tested in Chapter 5, the results are summarized in Table 6.3.

Table 6.2 Summary of resource vacations

| Vacation length (day) | Frequency | Count |
|---|---|---|
| 1 | 42 | 42 |
| 2 | 10 | 20 |
| 3 | 8 | 24 |
| 4 | 3 | 12 |
| 5 | 14 | 70 |
| 6 | 1 | 6 |
| 7 | 15 | 105 |
| 8 | 2 | 16 |
| 9 | 1 | 9 |
| 10 | 1 | 10 |
| 11 | 1 | 11 |
| 130 | 1 | 130 |
| Total | 99 | 455 |

Table 6.3 Summary of results on project makespan, number of task splitting, and computational time

| | Deterministic | Biased random sampling | Regret based random sampling |
|---|---|---|---|
| Task splitting is not allowed. | 86 weeks<br>0 split<br>(1.08 sec.) | 49.6 weeks<br>0 split<br>(512.84 sec.) | 49.6 weeks<br>0 split<br>(517.6 sec.) |
| Task splitting is allowed without control over number of task split | 68 weeks<br>34 splits<br>(0.31 sec.) | 40.8 weeks<br>39 splits<br>(128.17 sec.) | 40.8 weeks<br>40 splits<br>(128.6 sec.) |
| Task splitting is allowed with a control of number of task split | 68 weeks<br>32 splits<br>(1.85 sec.) | 40 weeks<br>40 splits<br>(1067.7 sec.) | 40.8 weeks<br>42 splits<br>(1122.4 sec.) |

It can be seen from the results that only the stochastic methods (BRS and RBRS) with task splitting can schedule all design tasks with the current headcount and complete the project within its deadline. All four methods have almost the same level of performance. The best result is generated by using biased random sampling with task splitting and a control on the number of task splits. The best schedule, including the assigned mode, starting time, and finish time of each design tasks, is listed in Table 6.4.

Table 6.4 The best schedule for the project

| Task name | Assigned mode | Starting Period | Finishing Period |
|---|---|---|---|
| 1. Hood Criteria | 1 | 1 | 30 |
| 2. Hood Latch (Feasibility) | 1 | 1 | 20 |
| 3. Hood Latch (Beta) | 1 | 1 | 30 |
| 4. Hood Outer (Alpha) | 1 | 31 | 110 |
| 5. Hood Outer (Beta/Prod) | 2 | 111 | 194 |
| 6. Hood Inner (Alpha) | 1 | 31 | 110 |

| Task name | Assigned mode | Starting Period | Finishing Period |
|---|---|---|---|
| 7. Hood Inner (Beta/Prod) | 2 | 111 | 194 |
| 8. Fender Criteria | 1 | 1 | 20 |
| 9. Fender Outer (LH) (Alpha) | 2 | 21 | 105 |
| 10. Fender Outer (LH) (Beta/Prod) | 1 | 106 | 194 |
| 11. Fender Extension (LH) (Alpha) | 2 | 21 | 70 |
| 12. Fender Extension (LH) (Beta/Prod) | 2 | 71 | 136 |
| 13. Fender Outer (RH) | 1 | 195 | 199 |
| 14. Fender Extension (RH) | 2 | 195 | 199 |
| 15. Grille Criteria | 2 | 1 | 20 |
| 16. Grille – Chev (Feasibility) | 1 | 21 | 95 |
| 17. Grille – Chev Base (Beta) | 1 | 96 | 199 |
| 18. Grille – Chev Uplevel (Beta) | 2 | 96 | 200 |
| 19. Grille – GMC (Beta) | 1 | 96 | 198 |
| 20. Front Bumper Criteria | 2 | 1 | 25 |
| 21. Bar – Front Bumper (Alpha) | 1 | 26 | 105 |
| 22. Bar – Front Bumper (Beta) | 2 | 106 | 179 |
| 23. Bracket – Front Bumper (LH) (Alpha) | 2 | 26 | 65 |
| 24. Bracket – Front Bumper (LH) (Beta) | 2 | 106 | 139 |
| 25. Bracket – Front Bumper (RH) | 2 | 140 | 144 |
| 26. Brace – Front Bumper (LH) (Alpha) | 1 | 26 | 65 |
| 27. Brace – Front Bumper (LH) (Beta) | 2 | 106 | 152 |
| 28. Brace – Front Bumper (RH) | 2 | 195 | 199 |
| 29. Cap – Front Bumper Bar – Chev (Beta) | 2 | 106 | 191 |
| 30. Air Dam – Front Bumper – Chev (Beta) | 1 | 106 | 179 |
| 31. Headlamp – Chev (LH) (Beta) | 2 | 106 | 159 |
| 32. Headlamp – Chev (RH) (Beta) | 1 | 160 | 164 |
| 33. Bracket – Headlamp (LH) (Beta) | 2 | 106 | 194 |
| 34. Bracket – Headlamp (RH) (Beta) | 1 | 195 | 199 |

# 7. CONCLUSIONS

This research has investigated the effect of allowing task splitting for multi-mode resource-constrained project scheduling problems under the presence of resource vacations. Project situations where task splitting is likely to benefit a schedule are identified through a set of parameters that characterize the degree of difficulty of a project as well as the structure of resource vacations.

The other accomplishment of this work is the development of the priority rule-based heuristic that includes a new parameter that captures the tightness of the resource situation, which directly influences the quality of the solution. This parameter is constructed and implemented in a priority rule-based heuristic with task splitting. Results from extensive computational experiments have shown promising performance, and have shown that the heuristic can be used to effectively schedule large projects with reasonable computational effort.

The results of this research point to the following research possibilities.

⇒ Extending the problem setting by relaxing an assumption: The case study in Chapter 6 demonstrates the industrial potential of this research. However, more realistic applications can be achieved by relaxing an assumption of the problem. This research assumes that task splitting is allowed without incurring additional setup or duration of the task being split. This assumption can be relaxed by introducing a parameter or a function that generates additional duration for every time a task is split. By including this parameter, a study can be conducted to determine the amount of additional duration as a function or a percentage of the original duration where task splitting will still benefit the scheduling process.

⇒ Developing more powerful Heuristics: More powerful heuristics can be developed by using metaheuristic methods such as simulated annealing, tabu search, or genetic algorithms. Metaheuristic methods can guarantee better performance by using the priority rule-based heuristic solutions developed in this research as initial solutions.

⇒ Including other types of resources: This research only considers one type of resource which are renewable resources. Another possible future research direction is to include other types of resources, including non-renewable resource, doubly-constrained resource, and/or partially renewable resource.

⇒ Considering the generalized precedence relations: More realistic problem settings can be modeled by including generalized precedence relations. These include precedence relations of the types, start-start, start-finish, finish-start, and finish-finish with a minimal time lag and a maximal time lag, as discussed in Chapter 2.

⇒ Constructing a lower bound for the solution: One aspect of this study that has not been included is a more effective way to evaluate the performance of the proposed heuristics. This can be achieved by constructing a good lower bound for optimal solutions. Possible approaches are Lagrangean relaxation or problem decompositions.

# BIBLIOGRAPHY

Alvarez-Valdes, R., and Tamarit, J., M. (1989) Heuristic Algorithms for Resource-Constrained Project Scheduling: A Review and An Empirical Analysis, *Advances in Project Scheduling*, Edited by Slowinski, R. and J. Weglarz, 113-134.

Bell, C., and Han, J. (1991), A New Heuristic Solution Method in Resource-Constrained Project Scheduling, *Naval Research Logistics*, 38, 315-331.

Bianco, L., Caramia, M., and Dell'Olmo, P. (1999) Solving a Preemptive Project Scheduling Problem with Coloring Techniques, *Project Scheduling: Recent Models, Algorithms, and Applications*, Edited by J. Weglarz, 135-145.

Bocter, F. (1990), Some Efficient Multi-Heuristic Procedures for Resource-Constrained Project Scheduling, *European Journal of Operational Research*, 49, 3-13.

Bocter, F. (1993), Heuristics for Scheduling Projects with Resource Restrictions and Several Resource-Duration Modes, *International Journal of Production Research*, 31 (11), 2547-2558.

Bottcher, J., Drexl, A., Kolisch, R., and Salewski, F. (1999), Project Scheduling Under Partially Renewable Resource Constraints, *Management Science*, 45 (4), 543-559.

Brucker, P., Drexl, A., Mohring, R., and Neumann, K. (1999), Invited Review: Resource-Constrained Project Scheduling: Notation, Classification, Models, and Methods, *European Journal of Operational Research*, 112, 3-41.

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

Christofides, N., and Alvarez-Valdes, R., Tamarit, J., M. (1987), Project Scheduling with Resource Constraints: A Branch and Bound Approach, *European Journal of Operational Research*, 29, 262-273.

Cooper, D. (1976), Heuristics for Scheduling Resource-Constrained Projects: An Experimental Investigation, *Management Science*, 22 (11), 1186-1194.

Davis, E., W., and Heidorn, G., E. (1971), An Algorithm for Optimal Project Scheduling under Multiple Resource Constraints, *Management Science*, 17 (12), 803–816.

Davis, E., and Patterson, J., H. (1975), A Comparison of Heuristic And Optimum Solutions in Resource-Constrained Project Scheduling, *Management Science*, 21 (8), 944-955.

Demeulemeester, E., L., and Herroelen, W., S. (1992), A Branch-and-Bound Procedure for the Multiple Resource-Constrained Project Scheduling Problems, *Management Science*, 38 (12), 1803-1818.

Demeulemeester, E., L., and Herroelen, W., S. (1995), New Benchmark Results for the Resource-Constrained Project Scheduling Problem, *Management Science*, 43 (11), 1485-1492.

Demeulemeester, E., L., and Herroelen, W., S. (1996), An Efficient Optimal Solution Procedure for the Preemptive Resource-Constrained Project Scheduling Problem, *European Journal of Operational Research*, 90, 334-338.

De Reyck, B., Demeulemeester, E., and Herroelen, W., S. (1999), Algorithms for Scheduling Projects with Generalized Precedence Relations, *Project Scheduling: Recent Models, Algorithms, and Applications*, Edited by J. Weglarz, 77-105.

De Reyck, B., and Herroelen, W., S. (1999), The Multi-Mode Resource-Constrained Project Scheduling Problem with Generalized Precedence Relations, *European Journal of Operational Research*, 119, 538-556.

Drexl, A. (1991), Scheduling of Project Networks by Job Assignment, *Management Science*, 37 (12), 1590-1602.

Drexl, A., Juretzka, J., Salewski, F., and Schirmer, A. (1999), New Modeling Concepts and Their Impact on Resource-Constrained Project Scheduling, *Project Scheduling: Recent Models, Algorithms, and Applications*, Edited by J. Weglarz, 413-432.

Drexl, A., Nissen, R., Patterson, J., H., and Salewski, F. (2000), ProGen/πx - An Instance Generator for Resource-Constrained Project Scheduling Problems with Partially Renewable Resources and Further Extensions, *European Journal of Operational Research*, 125, 59-72.

Garey, M., and Johnson, D. (1979), *Computer and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, New York.

Hartmann, S. (2001), Project Scheduling with Multiple Modes: A Genetic Algorithm, *Annals of Operations Research*, 102 (1), 111-135

Hartmann, S. (2002), A Self-Adapting Genetic Algorithm for Project Scheduling under Resource Constraints, *Naval research logistics*, 49 (5), 433-448

Hartmann, S., and Drexl, A. (1997), Project Scheduling with Multiple Modes: A Genetic Algorithm, Technical Report 435, *Manuskripte aus den Instituten fur Betriebswirtschaftslehre der Universitat Kiel*, Germany.

Hartmann, S., and Drexl, A. (1998), Project Scheduling with Multiple Modes: A Comparison of Exact Algorithms, *Networks*, 32, 283-297.

Herroelen, W., Demeulemeester, E., and De Reyck, B. (1999), A Classification Scheme for Project Scheduling, *Project Scheduling: Recent Models, Algorithms, and Applications*, Edited by J. Weglarz, 1-26.

Kaplan, L., A. (1988), Resource-Constrained Project Scheduling with Setup Times, *Unpublished Paper*, Department of Management, University of Tennessee.

Kelley, J., E., Jr. (1963), The Critical Path Method: Resource Planning and Scheduling, Industrial Scheduling, Edited by J. F. Muth and G. L. Thompson, 347-365.

Kolisch, R. (1996a), Serial and Parallel Resource-Constrained Project Scheduling Methods Revisited: Theory and Computation, *European Journal of Operational Research*, 90, 320-333.

Kolisch, R. (1996b), Efficient Priority Rules for the Resource-Constrained Project Scheduling Problem, *Journal of Operational Management*, 14, 179-192.

Kolisch, R., and Drexl, A. (1996), Adaptive Search for Solving Hard Project Scheduling Problems, *Naval Research Logistics,* 43, 23-40.

Kolisch, R., and Hartmann, S. (1999), Heuristic Algorithms for the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis, *Project Scheduling: Recent Models, Algorithms, and Applications*, Edited by J. Weglarz, 147-178.

Kolisch, R., Schwindt, C., and Sprecher, A. (1999), Benchmark Instances for Project Scheduling Problems, *Project Scheduling: Recent Models, Algorithms, and Applications*, Edited by J. Weglarz, 197-212.

Kolisch, R., and Sprecher, A. (1996), PSPLIB – A Project Scheduling Problem Library, *European Journal of Operational Research*, 96, 205-216.

Kolisch, R., Sprecher, A., and Drexl, A. (1992), Characterization and Generation of a General Class of Resource-Constrained Project Scheduling Problems: Easy and Hard Instances, Technical Report 301, *Manuskripte aus den Instituten fur Betriebswirtschaftslehre der Universitat Kiel*, Germany.

Kolisch, R., Sprecher, A., and Drexl, A. (1995), Characterization and Generation of a General Class of Resource-Constrained Project Scheduling Problems, *Management Science*, 41 (10), 1693-1703.

Leon, V., and Ramamoorthy, B. (1995), Strength and Adaptability of Problem-Space Based Neighborhoods for Resource-Constrained Scheduling, *OR Spektrum*, 17, 173-182.

McCullagh, P., and Nelder, J., A. (1989), *Generalized Linear Models*, Chapman & Hall/CRC.

Mingozzi, A., Maniezzo, V., Ricciardelli, S., and Bianco, L. (1998), An Exact Algorithm for the Resource-Constrained Project Scheduling Problem Based on a New Mathematical Formulation, *Management Science*, 44 (5), 714-729.

Oguz, O., and Bala, H. (1994), A Comparative Study of Computational Procedures for the Resource-Constrained Project Scheduling Problem, *European Journal of Operational Research*, 72, 406-416.

Ozdamar, L., and Ulusoy, G. (1995), A Survey on the Resource-Constrained Project Scheduling Problem, *IIE Transactions*, Vol. 27, 574-586.

Patterson, J., H. (1984), A Comparison of Exact Approaches for Solving the Multiple Constrained Resource, Project Scheduling Problems, *Management Science*, 30 (7), 854-867.

Patterson, J., H., Slowinski, R., Talbot, F., B., and Weglarz, J. (1989), An Algorithm for a General Class of Precedence and Resource Constrained Scheduling Problems, *Advances in Project Scheduling*, Edited by R. Slowinski and J. Weglarz, 3-28.

Patterson, J., H., Talbot, F., B., Slowinski, R., and Weglarz, J. (1990), Computational Experience with a Backtracking Algorithm for Solving a General Class of Precedence and Resource Constrained Scheduling Problems, *European Journal of Operational Research*, 49, 68-79.

Pollack-Johnson, B. (1995), Hybrid Structures and Improving Forecasting and Scheduling in Project Management, *Journal of Operational Management*, 12, 101-117.

Pritsker, A., Watters, L., and Wolfe, P. (1969), Multi-Project Scheduling with Limited Resources: A Zero-One Programming Approach, *Management Science*, 16 (1), 93-109.

Salewski, F., Schirmer, A., and Drexl, A. (1997), Project Scheduling Under Resource and Mode Identity Constraints: Model, Complexity, Methods, and Application, *European Journal of Operational Research*, 102, 88-110.

Sampson, S., and Weiss, E. (1993), Local Search Techniques for the Generalized Resource-Constrained Project Scheduling Problem, *Naval Research Logistics*, 40, 665-675.

Schirmer, A., and Drexl, A. (2001), Allocation of Partially Renewable Resources: Concept, Capabilities, and Application, *Networks*, 37 (1), 21-34.

Schirmer, A., and Riesenberg, S. (1997), Parameterized Heuristics for Project Scheduling – Biased Random Sampling Methods, Technical Report 456, *Manuskripte aus den Instituten fur Betriebswirtschaftslehre der Universitat Kiel*, Germany.

Shaffer, L., Ritter, J., and Meyer, W. (1965), *The Critical-Path Method*, McGraw Hill, New York.

Simpson, W., P. (1991), A Parallel Exact Solution Procedure for the Resource-Constrained Project Scheduling Problem, *Unpublished Ph.D. Dissertation*, Indiana University.

Simpson, W., P., and Patterson, J., H. (1996), A Multiple-Tree Search Procedure for the Resource-Constrained Project Scheduling Problem, *European Journal of Operational Research*, 89, 525-542.

Slowinski, R. (1981), Multiobjective Network Scheduling with Efficient Use of Renewable and Nonrenewable Resources, *European Journal of Operational Research*, 7, 265-273.

Sprecher, A., and Drexl, A. (1998), Multi-Mode Resource-Constrained Project Scheduling by a Simple, General and Powerful Sequencing Algorithm, *European Journal of Operational Research*, 107, 431-450.

Sprecher, A., Hartmann, S., and Drexl, A. (1997), An Exact Algorithm for Project Scheduling with Multiple Modes, *OR Spektrum*, 19, 195-203.

Sprecher, A., Kolisch, R., and Drexl, A. (1995), Semi-active, Active, and Non-delay Schedules for the Resource-Constrained Project Scheduling Problem, *European Journal of Operational Research*, 80, 94-102.

Stinson, J., Davis, E., and Khumawala, B. (1978), Multiple Resource-Constrained Scheduling Using Branch and Bound, *AIIE Transactions*, 10 (3), 252-259.

Talbot, B. (1982), Resource-Constrained Project Scheduling with Time-Resource Trade-offs: the Non-Preemptive Case, Management Science, 28 (10), 1197-1210.

Thomas, P., and Salhi, S. (1997), An Investigation into the Relationship of Heuristic Performance with Network-Resource Characteristics, *Journal of Operational Research Society*, 48 (1), 34-43.

Ulusoy, G. and Ozdamar, L. (1989), Heuristic Performance and Network/Resource Characteristics in Resource-Constrained Project Scheduling, *Journal of Operational Research Society*, 40 (12), 1145-1152.

Valls, V., Perez, M., and Quintanilla, M. (1992), Heuristic Performance in Large Resource-Constrained Projects, Technical Report 92-2, *Departament D'Estadistica I Invecigacio Operativa, Universitat de Valencia*, Germany.

Valls, V., Laguna, M., Lino, P., Perez, A., and Quintanilla, S. (1999), Project Scheduling with Stochastic Activity Interruptions, in: J. Weglarz (Ed.), Project Scheduling: Recent Models, Algorithms, and Applications, Kluwer Academic Publishers, Massachusetts, 333-353.

# APPENDICES

# APPENDIX A: STATISTICAL ANALYSIS FOR EXPERIMENT 1

## A.1 Summary of Statistical Finding

There is no statistical evidence that the variation between two levels of network complexity has significant effect on the probability of improvement on the solutions, after accounting for $RF$, $V_P$, $RS$, $V_L$, and $RS \times V_L$, p-value = 0.2125. However, there is a convincing evidence that $RF$, $V_P$, $RS$, and $V_L$ have significant effects on the probability of improvement on the solutions, p-value < 0.05.

The odds of improving the solutions for the resource factor of 1.0 were estimated to be 4.29 times the odds of improving the solutions for the resource factor of 0.5, after accounting for $V_P$, $RS$, $V_L$, and $RS \times V_L$, p-value < .0001. The odds of improving the solution for 20% resource vacations were estimated to be 5.25 times the odds of improving the solution for 10% resource vacations, after accounting for $RF$, $RS$, $V_L$, and $RS \times V_L$, p-value < .0001.

At resource strengths of 0.25 (0.75), the odds of improving the solutions for the short resource vacations ($V_L = 0$) were estimated to be 6.36 (31.72) times the odds of improving the solutions for the long resource vacation ($V_L = 1.0$), at the same levels of $RF$ and $V_P$, p-value = 0.0467. For the short (long) vacation, $V_L = 0$ (1.0), the odds of improving the solutions for the resource strength of 0.25 is estimated to be 10.99 (31.72) times the odds of improving the solutions for the resource strength of 0.75, at the same levels of $RF$ and $V_P$, p-value = 0.0467.

## A.2 Scope of Inference

The statistical findings stated above are from the results of a randomized experiment. Therefore, causal effects can be drawn and inference can be made to the class of MMRCPSP included in this experiment (10-task project with ProGen parameter as specified in Table 4.5). However, making inference beyond the scope of number of tasks and values of parameters used in this study is speculative.

## A.3 Details of Statistical Analysis

### Step 0: Recoding the responses

Response variable representing a measurement of the improvement on the schedule solution is defined as a percentage of improvement on the solution obtained with task preemption over the one without task preemption.

$$\%improvement = 100 \times \frac{makespan(no\_preemption) - makespan(preemption)}{makespan(no\_preemption)}.$$

For the purpose of this analysis, the responses are recoded as binary responses, where 0 indicates no improvement and 1 indicates improvement on the solution

### Step 1: Fitting a full model

The full model containing all five main factors and all 2-way interactions is fitted using generalized linear model with binomial link. The p-values of the experimental factors can be shown in Table A.1.

Table A.1 Significant level for each experimental factor

| Experimental factor | p-value |
|---|---|
| $NC$ | 0.2125 |
| $RF$ | **< 0.0001** |
| $RS$ | **< 0.0001** |
| $V_P$ | **< 0.0001** |
| $V_L$ | **< 0.0001** |
| $NC \times RF$ | 0.3594 |
| $NC \times RS$ | 0.7208 |
| $NC \times V_P$ | 0.6108 |
| $NC \times V_L$ | 0.5211 |
| $RF \times RS$ | 0.1754 |
| $RF \times V_P$ | 0.4766 |
| $RF \times V_L$ | 0.5621 |
| $RS \times V_P$ | 0.9263 |
| $RS \times V_L$ | **0.0223** |
| $V_P \times V_L$ | 0.8040 |

It can be seen from Table A.1 that 1) among the five main effects, network complexity is not significant, p-value = 0.2125, whereas all other four main effects are significant with p-values less than 0.0001, 2) among all ten 2-way interactions only the interaction between resource factor and resource vacation length $RS \times V_L$ is moderately significant, p-value = 0.0223.

Step 2: Fitting a reduced model

In order to estimate the effect of each significant factor and interaction, a reduced model is fitted. Significances of the interaction terms are checked using Drop-in-deviance p-values as given in Table A.2.

Table A.1 Drop-in-Deviance p-values

| Test for significance of | Drop in Deviance p-value | At 0.05 level of significance |
|---|---|---|
| $RS\times V_P, V_P\times V_L, NC\times RS$ | 0.9789 | No evidence of these interactions |
| $RF\times V_L, NC\times V_P, NC\times V_L$ | 0.7716 | No evidence of these interactions |
| $NC\times RF, RF\times V_P, RF\times RS$ | 0.2627 | No evidence of these interactions |
| $NC$ | 0.2044 | No evidence of this main effect |
| $RS\times V_L$ | 0.0467 | Significant evidence of this interaction |
| $RF$ | <.0001 | Significant evidence of this main effect |
| $RS$ | <.0001 | Significant evidence of this main effect |
| $V_P$ | <.0001 | Significant evidence of this main effect |
| $V_L$ | <.0001 | Significant evidence of this main effect |

Drop-in-deviance p-values show significance of four main effects and one interaction ($RF, RS, V_P, V_L$, and $RS\times V_L$). Those are included in the final model.

Step 3: Final model results

The final model is,

$$\log\left(\frac{\pi_i}{1-\pi_i}\right)=0.36+1.46RF-2.40RS+1.66V_P-1.85V_L+0.79(RS\times V_L) \qquad (A.1)$$

where $RF$, $RS$, $V_P$, and $V_L$ are all binary indicator variables, which take a value of 1 when the factor is at level 2 (according to Table 4.5), and 0 otherwise.

The results of the analysis are shown in the following tables.

Table A.3 Criteria for assessing Goodness of Fit

| Criterion | DF | Value | Value/DF |
|---|---|---|---|
| Deviance | 634 | 635.2392 | 1.0020 |
| Scaled Deviance | 634 | 635.2392 | 1.0020 |
| Pearson Chi-Square | 634 | 594.6523 | 0.9379 |
| Scaled Pearson $\chi^2$ | 634 | 594.6523 | 0.9379 |
| Log Likelihood | | -317.6196 | |

Table A.4 Estimate of significant factors

| Parameter | DF | Estimate | S.E. | Wald 95% C.I. | | $\chi^2$ | Pr > $\chi^2$ |
|---|---|---|---|---|---|---|---|
| Intercept | 1 | 0.3559 | 0.2426 | -0.1196 | 0.8314 | 2.15 | 0.1424 |
| $RF$ | 1 | 1.4571 | 0.2055 | 1.0542 | 1.8599 | 50.25 | <.0001 |
| $RS$ | 1 | -2.3959 | 0.3016 | -2.9869 | -1.8048 | 63.11 | <.0001 |
| $V_P$ | 1 | 1.6590 | 0.2081 | 1.2511 | 2.0670 | 63.54 | <.0001 |
| $V_L$ | 1 | -1.8503 | 0.2945 | -2.4275 | -1.2732 | 39.48 | <.0001 |
| $RS \times V_L$ | 1 | 0.7893 | 0.3985 | 0.0083 | 1.5702 | 3.92 | 0.0476 |

Step 4: Estimate odds ratios

The odds ratios given in the statistical findings are estimated by $e^{\beta}$, where $\beta$ is the coefficient of the factor of interest. For instance, the odds ratios of improving the solution for $RF$ of 1.0 to $RF$ of 0.5 are estimated to be $e^{1.4571} = 4.2935$. Thus, estimates of odds ratio for other experimental factors can be performed similarly.

# APPENDIX B: STATISTICAL ANALYSIS FOR EXPERIMENT 2

## B.1 Summary of Statistical Finding

Given that task splitting improves the solutions, there is no statistical evidence that the variation between three levels of resource factor has significant effect on the degree of improvement on the solutions, after accounting for $TASK$, $V_P$, $RS$, $v_l$, and $RS \times V_P$, p-value = 0.0814. However, there is a convincing evidence that $TASK$, $V_P$, $RS$, and $V_L$ have significant effects on the degree of improvement on the solutions.

Given the task splitting improves the solutions, changing the resource vacation length from long to medium increases the median of percent improvement by 29.69% with a 95% confidence interval of (19.52%, 40.73%), after adjusting for $TASK$, $V_P$, and $RS$, (49.18% with a 95% confidence interval of (37.37%,62.02%) for from long to short).

Table B.1 Effects of resource strength depending on percent resource vacations

| Each cell is a decrease in median of percent improvement by | | changing $RS$ from 0.25 to | | |
|---|---|---|---|---|
| | | 0.50 | 0.75 | 1.0 |
| at $V_P$ of | 20% | 32.97% (23.45%,41.31%) | 40.55% (31.37%,48.50%) | 50.84% (37.88%,61.09%) |
| | 30% | 40.55% (32.89%,47.33%) | 50.34% (43.71%,56.19%) | 57.68% (50.27%,63.99%) |

For resource strength, changing from 0.25 to other levels would decrease the mean of percent improvement, depending on the level of percent resource vacations. Table B.1 summarizes the percent decrease in the median of percent improvement for each

combination of resource strength and percent resource vacations with a 95% confidence interval in parentheses underneath.

Table B.2 Effects of percent resource vacations depending on resource strength

| Each cell is a increase in median of percent improvement by | | changing $V_P$ from 10% to 20% | changing $V_P$ from 10% to 30% |
|---|---|---|---|
| at $RS$ of | 0.50 | 36.34% (16.48%,59.60%) | 103.40% (74.87%,136.58%) |
| | 0.75 | 18.53% (-1.45%,42.56%) | 66.53% (40.00%,98.08%) |
| | 1.0 | 13.88% (-27.53%,78.96%) | 64.87% (8.30%,150.99%) |

For percent resource vacations, changing from 10% to other levels would increase the median of percent improvement, depending on the level of resource strength. Table B.2 summarizes the percent in the median of percent improvement for each combination of resource strength and percent resource vacations with a 95% confidence interval in parentheses underneath. It can be noticed that changing $V_P$ from 10% to 20% results in inconclusive changes in median of percent improvement when $RS = 0.75$ and 1.0, since the confidence intervals contain zeros.

It must be note that even though the effect of the number of task included in the project is significant, its effect is not of interest for this study. Thus, there is no need to interpret the effect of $TASK$. The significance of this categorical variable helps interpreting the effect of other factors of interest in because its effect is accounted for in the statistical analysis.

## B.2 Scope of Inference

The statistical inference of the results is made from a randomized experiment. Thus, causal effects can be drawn for this study. However, statistical inference can be made only to the class of MMRCPSP included in this experiment (8-13 tasks project with ProGen parameter as specified in Table 4.10). Making inference beyond the scope of number of tasks and values of parameters used in this experiment is speculative.

## B.3 Details of Statistical Analysis

### Step 0: Dealing with response variable

For the purpose of this experiment, the responses with no improvement are excluded from the analysis. This makes the data unbalanced. Fortunately, the statistical procedure used in this analysis can handle unbalanced data. The estimates of the regression parameters are still unbiased and the p-values are still valid. In addition, the analysis is performed on the responses with the log-scale percent improvement in order to satisfy the model assumptions. Residual plots and normal probability plot for the final model are used to validate the assumptions. The plots are given at the end of this section.

### Step 2: Fitting a full model

The full model containing all four main factors, a number of tasks as categorical variable, and all ten possible combinations of 2-way interactions is fitted. The p-values of the factors included in the full model are as shown in Table B.3.

Table B.3 Significance level of experimental factors

| Experimental factors | p-value |
|---|---|
| $TASK$ | <.0001 |
| $RF$ | 0.5271 |
| $RS$ | <.0001 |
| $V_P$ | <.0001 |
| $V_L$ | <.0001 |
| $TASK \times RF$ | 0.6153 |
| $TASK \times RS$ | 0.7540 |
| $TASK \times V_P$ | 0.9129 |
| $TASK \times V_L$ | 0.2962 |
| $RF \times RS$ | 0.4614 |
| $RF \times V_P$ | 0.5405 |
| $RF \times V_L$ | 0.6057 |
| $RS \times V_P$ | <.0001 |
| $RS \times V_L$ | 0.0699 |
| $V_P \times V_L$ | 0.0373 |

Step 3: Fitting a reduced model

Similar to Appendix A, a reduced model is fitted to estimate each significant factor effect. Drop-in-deviance p-values are used in the model selection process, resulting in the final model as shown in (B.1), which includes three main effects, one interaction, and the number of tasks ( $RS, V_P, V_L$, $RS \times V_P$, and $TASK$ ).

$$\begin{aligned}\log(\%improvement) &= 2.32 - 0.05TASK - 0.16RS_{0.5} - 0.14RS_{0.75} \\ &- 0.29RS_{1.0} + 0.55V_{P,20\%} + 1.07V_{P,30\%} + 0.26V_{L,medium} + 0.40V_{L,short} \\ &- 0.24RS_{0.5} \times V_{P,20\%} - 0.36RS_{0.5} \times V_{P,30\%} - 0.38RS_{0.75} \times V_{P,20\%} \\ &- 0.56RS_{0.75} \times V_{P,30\%} - 0.42RS_{1.0} \times V_{P,20\%} - 0.57RS_{1.0} \times V_{P,30\%}\end{aligned} \quad \text{(B.1)}$$

where $RS$, $V_P$, and $V_L$ are all binary indicator variables, which take a value of 1 when the factor is at level 2 (according to Table 4.10), and 0 otherwise, and $TASK$ is categorical variable, ranging from 8 to 13. Table B.4 and B.5 show the ANOVA table, and type I p-values and type III p-values, respectively.

Table B.4 Analysis of variance

| Source | DF | Sum of Squares | Mean Square | F Value | Pr > F |
|---|---|---|---|---|---|
| Model | 14 | 293.4796 | 20.9628 | 46.68 | <.0001 |
| Error | 1670 | 749.9733 | 0.4491 | | |
| Corrected Total | 1684 | 1043.4528 | | | |

Table B.5 Type I and Type III p-values

| Source | DF | Type I SS | Mean Square | F Value | Pr > F |
|---|---|---|---|---|---|
| $TASK$ | 1 | 11.9821 | 11.9821 | 26.68 | <.0001 |
| $RS$ | 3 | 72.5507 | 24.1836 | 53.85 | <.0001 |
| $V_P$ | 2 | 154.2369 | 77.1184 | 171.72 | <.0001 |
| $V_L$ | 2 | 41.5726 | 20.7863 | 46.29 | <.0001 |
| $RS \times V_P$ | 6 | 13.1373 | 2.1895 | 4.88 | <.0001 |

| Source | DF | Type III SS | Mean Square | F Value | Pr > F |
|---|---|---|---|---|---|
| $TASK$ | 1 | 13.4387 | 13.4387 | 29.92 | <.0001 |
| $RS$ | 3 | 65.9112 | 21.9704 | 48.92 | <.0001 |
| $V_P$ | 2 | 75.0419 | 37.5209 | 83.55 | <.0001 |
| $V_L$ | 2 | 41.5749 | 20.7874 | 46.29 | <.0001 |
| $RS \times V_P$ | 6 | 13.1373 | 2.1895 | 4.88 | <.0001 |

Table B.6 Estimates of significant factors

| Parameter | Estimate | Standard Error | t Value | Pr > \|t\| | 95% Confidence Lower Limits | 95% Confidence Upper Limits |
|---|---|---|---|---|---|---|
| Intercept | 2.3175 | 0.1196 | 19.38 | <.0001 | 2.0830 | 2.5521 |
| *TASK* | -0.0524 | 0.0096 | -5.47 | <.0001 | -0.0711 | -0.0336 |
| $RS_{1.0}$ | -0.2894 | 0.2096 | -1.38 | 0.1677 | -0.7006 | 0.1218 |
| $RS_{0.75}$ | -0.1356 | 0.0924 | -1.47 | 0.1424 | -0.3169 | 0.0457 |
| $RS_{0.5}$ | -0.1574 | 0.0831 | -1.89 | 0.0585 | -0.3204 | 0.0056 |
| $V_{P,30\%}$ | 1.0735 | 0.0693 | 15.49 | <.0001 | 0.9375 | 1.2094 |
| $V_{P,20\%}$ | 0.5526 | 0.0710 | 7.78 | <.0001 | 0.4132 | 0.6919 |
| $V_{L,short}$ | 0.3999 | 0.0416 | 9.6 | <.0001 | 0.3182 | 0.4815 |
| $V_{L,medium}$ | 0.2565 | 0.0421 | 6.1 | <.0001 | 0.1740 | 0.3391 |
| $RS_{1.0} \times V_{P,30\%}$ | -0.5743 | 0.2251 | -2.55 | 0.0108 | -1.0158 | -0.1328 |
| $RS_{1.0} \times V_{P,20\%}$ | -0.4181 | 0.2411 | -1.73 | 0.0831 | -0.8910 | 0.0549 |
| $RS_{0.75} \times V_{P,30\%}$ | -0.5593 | 0.1123 | -4.98 | <.0001 | -0.7797 | -0.3390 |
| $RS_{0.75} \times V_{P,20\%}$ | -0.3769 | 0.1179 | -3.2 | 0.0014 | -0.6081 | -0.1458 |
| $RS_{0.5} \times V_{P,30\%}$ | -0.3565 | 0.1036 | -3.44 | 0.0006 | -0.5597 | -0.1533 |
| $RS_{0.5} \times V_{P,20\%}$ | -0.2427 | 0.1072 | -2.26 | 0.0237 | -0.4530 | -0.0324 |

Step 4: Estimating factor effects without interaction

In the final model, significant factor that does not interact with other factors is the resource vacation length, $V_L$. This factor effect is additive on a log-scale of the response. Performing back-transformations on this factor yields multiplicative effects of the factor on the original scale. More specifically, changing the resource vacation length from long to medium will increase the mean of the log (percent improvement) by 0.26 (i.e. the coefficient of the term $V_{L,medium}$). Equivalently, this effect can be estimated on the original scale as, changing the resource vacation length from long to medium will increase the

median of percent improvement by a multiplicative factor of $e^{0.26} = 1.30$ (i.e. increase of 30%). Table B.6 shows estimates of the coefficients for significant factors in the final model.

Step 5: Estimating factor effects with interaction

Factor effects that interact with each other in the final model are resource strength, $RS$, and percent resource vacations, $V_P$. Because the interaction between the two factors is significant, these factor effects are estimated marginally by combining two coefficients involving both factors. For example, changing the resource strength from 0.25 to 0.5 while holding the percent resource vacation constant at 20% would decrease the mean of the log (percent improvement) by $-0.16 - 0.24 = -0.40$ (i.e. the sum of the coefficients for $RS_{0.5}$ and $RS_{0.5} \times V_{P,20\%}$). The decrease in the mean of the log (percent improvement) is equivalent to a multiplicative decrease of $e^{(-0.16-0.24)} = e^{-0.40} = 0.67$ in the median of the percent improvement. This means that the median of the percent improvement at $RS$ = 0.50 and $V_P$ = 20% would be 0.67 times the median of the percent improvement at $RS$ = 0.25 and $V_P$ = 20% (i.e. 33% decrease in median percent improvement). Thus, estimates of factor effects can be performed similarly for other experimental factors.

Step 6: Orthogonal contrasts estimations

To help understand the effect of the main factors with interactions with more than two levels, each of the main factors and the interactions are separated into orthogonal contrasts of mean effects, linear effects, quadratic effects, and cubic effects (only for $RS$ that has 4 levels). These effects are estimated by orthogonal contrasts with coefficients as given in Table B.7. The significance of each contrasts are given in Table B.8.

Table B.7 Orthogonal contrast estimation coefficients

| Orthogonal contrasts | Coefficients used for estimation |
|---|---|
| *linear* (2 levels) | -1 0 1 |
| *quadratic* (2 levels) | 1 -2 1 |
| *linear* (3 levels) | -3 -1 -1 3 |
| *quadratic* (3 levels) | 1 -1 -1 1 |
| *cubic* (3 levels) | -1 3 -3 1 |
| *linear*×*linear* | 3 0 -3 1 0 -1 -1 0 1 -3 0 3 |
| *quadratic*×*linear* | -1 0 1 1 0 -1 1 0 -1 -1 0 1 |
| *cubic*×*linear* | 1 0 -1 -3 0 3 3 0 -3 -1 0 1 |
| *linear*×*quadratic* | -3 6 -3 -1 2 -1 1 -2 1 3 -6 3 |
| *quadratic*×*quadratic* | 1 -2 1 -1 2 -1 -1 2 -1 1 -2 1 |
| *cubic*×*quadratic* | -1 2 -1 3 -6 3 -3 6 -3 1 -2 1 |

Table B.8 Estimates of orthogonal contrasts

| Parameter | Estimate | Standard Error | t Value | Pr > \|t\| | 95% Confidence Lower Limits | 95% Confidence Upper Limits |
|---|---|---|---|---|---|---|
| $RS_{linear}$ | 1.9511 | 0.2594 | 7.52 | <.0001 | 1.4422 | 2.4599 |
| $RS_{quadratic}$ | 0.1847 | 0.0969 | 1.91 | 0.0568 | -0.0054 | 0.3747 |
| $RS_{cubic}$ | 0.3483 | 0.1637 | 2.13 | 0.0336 | 0.0271 | 0.6694 |
| $V_{P,linear}$ | -0.7009 | 0.0635 | -11.03 | <.0001 | -0.8255 | -0.5763 |
| $V_{P,quadratic}$ | 0.1147 | 0.0951 | 1.21 | 0.228 | -0.0718 | 0.3012 |
| $V_{L,linear}$ | -0.3999 | 0.0416 | -9.6 | <.0001 | -0.4815 | -0.3182 |
| $V_{L,quadratic}$ | -0.1132 | 0.0686 | -1.65 | 0.0992 | -0.2478 | 0.0214 |
| $RS_{linear} \times V_{P,linear}$ | -1.9257 | 0.6854 | -2.81 | 0.005 | -3.2701 | -0.5813 |
| $RS_{quadratic} \times V_{P,linear}$ | -0.3415 | 0.2538 | -1.35 | 0.1786 | -0.8394 | 0.1563 |
| $RS_{cubic} \times V_{P,linear}$ | 0.0342 | 0.4178 | 0.08 | 0.9348 | -0.7852 | 0.8535 |
| $RS_{linear} \times V_{P,quadratic}$ | -0.8513 | 1.0023 | -0.85 | 0.3958 | -2.8172 | 1.1146 |
| $RS_{quadratic} \times V_{P,quadratic}$ | -0.0616 | 0.3802 | -0.16 | 0.8713 | -0.8073 | 0.6841 |
| $RS_{cubic} \times V_{P,quadratic}$ | -0.0649 | 0.6645 | -0.1 | 0.9222 | -1.3682 | 1.2385 |

Step 7: Calculating 95% confidence intervals for the factor effects

A 95% confidence intervals for the effect of short vacation length and medium vacation length, are $e^{(0.28\pm1.96\times0.0416)} = (1.2183, 1.4369)$, and $e^{(0.42\pm1.96\times0.0421)} = (1.4026, 1.6515)$, respectively, where 0.0416 and 0.0421 are the standard error of $V_{L,short}$ and $V_{L,medium}$, respectively (from Table B.6).

A 95% confidence intervals for the effects of resource strength and percent resource vacation are more complicated to calculate, because they interact. Formally, it can be calculated by $e^{(\beta_i+\beta_j \pm 1.96\times SE(\beta_i+\beta_j))}$, where $\beta_i$ and $\beta_j$ are the two coefficients that constitute the effect, and $SE(\beta_i+\beta_j)$ is the standard error of the sum of the coefficients. This standard error is calculated from the variance-covariance matrix given in Table B.9.

Table B.9 Variance-covariance matrix for coefficients in the final model

| Parameter | Intercept | *TASK* | $RS_{1.0}$ | $RS_{0.75}$ | $RS_{0.5}$ |
|---|---|---|---|---|---|
| Intercept | 1.4300E-02 | -9.8000E-04 | -3.2000E-03 | -3.0500E-03 | -3.2000E-03 |
| *TASK* | -9.8000E-04 | 9.2000E-05 | 4.3000E-05 | 9.8000E-06 | 1.5000E-05 |
| $RS_{1.0}$ | -3.2000E-03 | 4.3000E-05 | 4.3950E-02 | 3.0120E-03 | 2.9950E-03 |
| $RS_{0.75}$ | -3.0500E-03 | 9.8000E-06 | 3.0120E-03 | 8.5430E-03 | 2.9920E-03 |
| $RS_{0.5}$ | -3.2000E-03 | 1.5000E-05 | 2.9950E-03 | 2.9920E-03 | 6.9060E-03 |
| $V_{P,30\%}$ | -3.1700E-03 | 9.5160E-06 | 2.9800E-03 | 2.9900E-03 | 3.0010E-03 |
| $V_{P,20\%}$ | -3.0900E-03 | 4.4950E-06 | 2.9860E-03 | 2.9920E-03 | 2.9980E-03 |
| $V_{L,short}$ | -1.1500E-03 | 7.3800E-06 | -2.1000E-04 | -7.0000E-05 | 9.0000E-05 |
| $V_{L,medium}$ | -1.1000E-03 | 3.4490E-06 | -4.4000E-04 | -6.0000E-05 | 3.5000E-05 |
| $RS_{1.0}\times V_{P,30\%}$ | 3.0370E-03 | -2.0000E-05 | -4.3930E-02 | -3.0000E-03 | -3.0000E-03 |
| $RS_{1.0}\times V_{P,20\%}$ | 2.7560E-03 | 5.0000E-06 | -4.3930E-02 | -3.0000E-03 | -3.0000E-03 |
| $RS_{0.75}\times V_{P,30\%}$ | 3.0030E-03 | -2.4700E-06 | -3.0000E-03 | -8.5400E-03 | -2.9900E-03 |

| | | | | | |
|---|---|---|---|---|---|
| $RS_{0.5} \times V_{P,30\%}$ | 3.1830E-03 | -1.0000E-05 | -2.9900E-03 | -2.9900E-03 | -6.9100E-03 |
| $RS_{0.5} \times V_{P,20\%}$ | 3.0510E-03 | 2.2930E-06 | -2.9800E-03 | -2.9900E-03 | -6.9100E-03 |

| Parameter | $V_{P,30\%}$ | $V_{P,20\%}$ | $V_{L,short}$ | $V_{L,medium}$ |
|---|---|---|---|---|
| Intercept | -3.1700E-03 | -3.0900E-03 | -1.1500E-03 | -1.1000E-03 |
| *TASK* | 9.5160E-06 | 4.4950E-06 | 7.3800E-06 | 3.4490E-06 |
| $RS_{1.0}$ | 2.9800E-03 | 2.9860E-03 | -2.1000E-04 | -4.4000E-04 |
| $RS_{0.75}$ | 2.9900E-03 | 2.9920E-03 | -7.0000E-05 | -6.0000E-05 |
| $RS_{0.5}$ | 3.0010E-03 | 2.9980E-03 | 9.0000E-05 | 3.5000E-05 |
| $V_{P,30\%}$ | 4.8050E-03 | 2.9990E-03 | 1.0600E-04 | 8.3000E-05 |
| $V_{P,20\%}$ | 2.9990E-03 | 5.0470E-03 | 6.7000E-05 | 4.6000E-05 |
| $V_{L,short}$ | 1.0600E-04 | 6.7000E-05 | 1.7340E-03 | 1.0250E-03 |
| $V_{L,medium}$ | 8.3000E-05 | 4.6000E-05 | 1.0250E-03 | 1.7700E-03 |
| $RS_{1.0} \times V_{P,30\%}$ | -4.7900E-03 | -2.9900E-03 | 2.2000E-05 | 3.7600E-04 |
| $RS_{1.0} \times V_{P,20\%}$ | -2.9800E-03 | -5.0400E-03 | 6.9000E-05 | 4.2800E-04 |
| $RS_{0.75} \times V_{P,30\%}$ | -4.8000E-03 | -2.9900E-03 | 2.4000E-05 | 2.0000E-05 |
| $RS_{0.75} \times V_{P,20\%}$ | -2.9900E-03 | -5.0400E-03 | 4.1000E-05 | 6.9000E-05 |
| $RS_{0.5} \times V_{P,30\%}$ | -4.8000E-03 | -3.0000E-03 | -9.0000E-05 | -5.0000E-05 |
| $RS_{0.5} \times V_{P,20\%}$ | -3.0000E-03 | -5.0500E-03 | -1.4000E-04 | -7.0000E-05 |

| Parameter | $RS_{1.0} \times V_{P,30\%}$ | $RS_{1.0} \times V_{P,20\%}$ | $RS_{0.75} \times V_{P,30\%}$ | $RS_{0.75} \times V_{P,20\%}$ |
|---|---|---|---|---|
| Intercept | 3.0370E-03 | 2.7560E-03 | 3.0030E-03 | 2.8960E-03 |
| *TASK* | -2.0000E-05 | 5.0000E-06 | -2.4700E-06 | 5.2680E-06 |
| $RS_{1.0}$ | -4.3930E-02 | -4.3930E-02 | -3.0000E-03 | -3.0100E-03 |
| $RS_{0.75}$ | -3.0000E-03 | -3.0000E-03 | -8.5400E-03 | -8.5400E-03 |
| $RS_{0.5}$ | -3.0000E-03 | -3.0000E-03 | -2.9900E-03 | -2.9900E-03 |
| $V_{P,30\%}$ | -4.7900E-03 | -2.9800E-03 | -4.8000E-03 | -2.9900E-03 |
| $V_{P,20\%}$ | -2.9900E-03 | -5.0400E-03 | -2.9900E-03 | -5.0400E-03 |
| $V_{L,short}$ | 2.2000E-05 | 6.9000E-05 | 2.4000E-05 | 4.1000E-05 |
| $V_{L,medium}$ | 3.7600E-04 | 4.2800E-04 | 2.0000E-05 | 6.9000E-05 |
| $RS_{1.0} \times V_{P,30\%}$ | 5.0680E-02 | 4.3940E-02 | 4.8000E-03 | 3.0070E-03 |
| $RS_{1.0} \times V_{P,20\%}$ | 4.3940E-02 | 5.8140E-02 | 2.9970E-03 | 5.0610E-03 |
| $RS_{0.75} \times V_{P,30\%}$ | 4.8000E-03 | 2.9970E-03 | 1.2620E-02 | 8.5390E-03 |

| | | | | |
|---|---|---|---|---|
| $RS_{0.75} \times V_{P,20\%}$ | 3.0070E-03 | 5.0610E-03 | 8.5390E-03 | 1.3890E-02 |
| $RS_{0.5} \times V_{P,30\%}$ | 4.8010E-03 | 2.9920E-03 | 4.7970E-03 | 2.9910E-03 |
| $RS_{0.5} \times V_{P,20\%}$ | 2.9950E-03 | 5.0430E-03 | 2.9920E-03 | 5.0420E-03 |

| Parameter | $RS_{0.5} \times V_{P,30\%}$ | $RS_{0.5} \times V_{P,20\%}$ |
|---|---|---|
| Intercept | 3.1830E-03 | 3.0510E-03 |
| *TASK* | -1.0000E-05 | 2.2930E-06 |
| $RS_{1.0}$ | -2.9900E-03 | -2.9800E-03 |
| $RS_{0.75}$ | -2.9900E-03 | -2.9900E-03 |
| $RS_{0.5}$ | -6.9100E-03 | -6.9100E-03 |
| $V_{P,30\%}$ | -4.8000E-03 | -3.0000E-03 |
| $V_{P,20\%}$ | -3.0000E-03 | -5.0500E-03 |
| $V_{L,short}$ | -9.0000E-05 | -1.4000E-04 |
| $V_{L,medium}$ | -5.0000E-05 | -7.0000E-05 |
| $RS_{1.0} \times V_{P,30\%}$ | 4.8010E-03 | 2.9950E-03 |
| $RS_{1.0} \times V_{P,20\%}$ | 2.9920E-03 | 5.0430E-03 |
| $RS_{0.75} \times V_{P,30\%}$ | 4.7970E-03 | 2.9920E-03 |
| $RS_{0.75} \times V_{P,20\%}$ | 2.9910E-03 | 5.0420E-03 |
| $RS_{0.5} \times V_{P,30\%}$ | 1.0730E-02 | 6.9060E-03 |
| $RS_{0.5} \times V_{P,20\%}$ | 6.9060E-03 | 1.1500E-02 |

For instance, a 95% confidence interval for the effect of changing the resource strength from 0.25 to 0.5, while holding percent resource vacation fixed at 20% is $e^{(-0.16-0.24\pm1.96\times0.0677)} = (0.5869, 0.7655)$, where the standard error of 0.0677 is calculated from $\sqrt{(6.9060\times10^{-3})+(1.1500\times10^{-2})+2(-6.9100\times10^{-3})} = 0.0677$. In other words, the multiplicative effect of $e^{(-0.16-0.24)} = e^{-0.40} = 0.67$ has a 95% confidence interval of (0.59,0.77), which implies that the $100(1-0.67)\% = 33\%$ decrease in median of percent improvement has a 95% confidence interval of 23% and 41%.

Step 8: Checking model assumptions

Step 8: Checking model assumptions

Two plots mentioned in Step 1 are shown here. It can be seen that the residual plots seem to be in a random pattern, and the normal quantile probability plot shows no sign of non-normality. Therefore, model assumptions are satisfied.

Figure B.1 Residual plots of residuals vs. predicted mean of log (percent improvement)

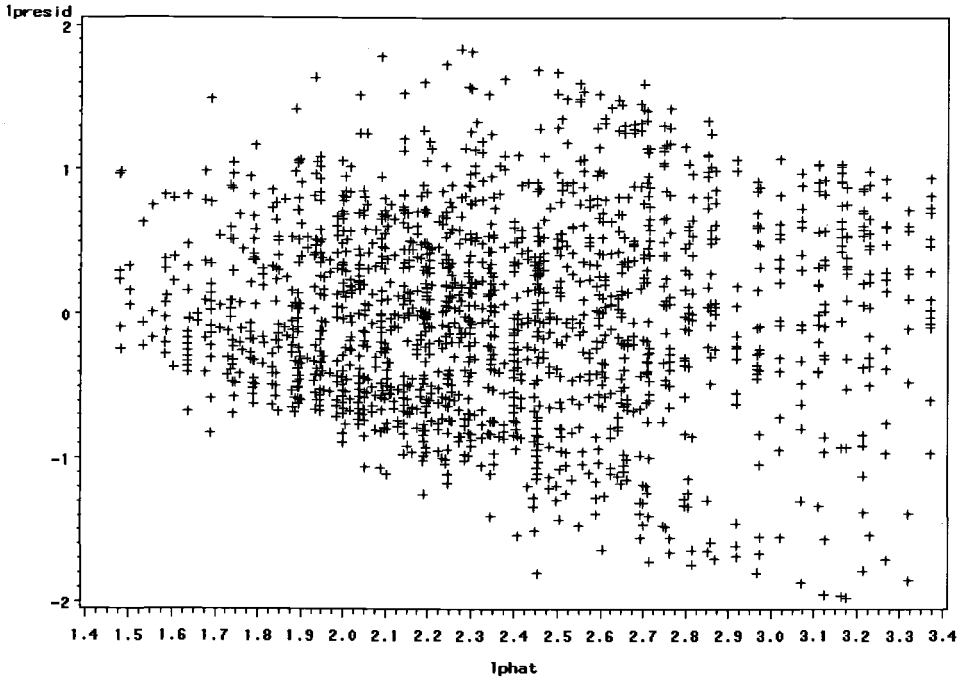


Figure B.2 Normal probability plot