Abstract

RAMACHANDRA, GIRISH A. Optimal Dynamic Resource Allocation in Activity Networks. (Under the direction of Professor Salah E. Elmaghraby).

We treat the problem of optimally allocating resources of limited availability under uncertainty to the various activities of a project to minimize a certain economic objective composed of resource cost and tardiness cost. Traditional project scheduling methods assume that the uncertainty resides in the duration of the activities. Our research assumes that the work content (or "effort") of an activity is the source of uncertainty and the duration is the result of the amount of resource allocated to the activity, which then becomes the decision variable. The functional relationship between the work content (w), the resource allocation (x), and the duration of the activity (y) is arbitrary, though we assume that the relationship obeys the "power law." In other words, $y = f(w, x^{\gamma})$, where the exponent γ is some constant.

As preliminary, we first treat the problem assuming that the work content is known deterministically. We develop two new models, a nonlinear programming model, which can be used when resource availabilities are continuous, and an integer program that handles the case when resource availabilities are discrete. When the work content is known only in probability, we first treat the special case when the work content is exponentially distributed. This results in a continuous-time Markov chain with a single absorbing state. We establish convexity of the cost function and develop a Policy Iteration–like approach that achieves the optimum in a finite number of steps. In case of arbitrary probability distribution of the work content, we develop a simulation-cum optimization method that incorporates sampling optimization and variance reduction techniques, and which can be used for the purposes of estimation of total project cost, resource consumption levels, etc.

OPTIMAL DYNAMIC RESOURCE ALLOCATION IN ACTIVITY NETWORKS

BY

GIRISH A. RAMACHANDRA

A DISSERTATION SUBMITTED TO THE GRADUATE FACULTY OF
NORTH CAROLINA STATE UNIVERSITY
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

INDUSTRIAL ENGINEERING

RALEIGH
MAY 2006

APPROVED BY:

DR. SUBHASHIS GHOSAL	DR. MATTHIAS F. STALLMANN
DR. ABDELHAKIM ARTIBA EXTERNAL COMMITTEE MEMBER	DR. JAMES R. WILSON

DR. SALAH E. ELMAGHRABY
CHAIR OF ADVISORY COMMITTEE

To my parents, Dr. R. N. Bhat and Smt. Shantha R. Bhat.

Biography

Girish Ramachandra was born on February 2nd, 1974 in Bangalore, India. He earned his Bachelors degree in Mechanical Engineering in 1995 from National Institute of Technology (formerly known as Karnataka Regional Engineering College) at Suratkal, India.

He then joined Larsen & Toubro group of companies, Mumbai, India, where he worked in their earth moving equipment manufacturing division for four years in production planning and materials control.

In 1999, he joined North Carolina State University to pursue his graduate studies in Industrial Engineering, where he received his Masters degree in Industrial Engineering in 2002. During the period August 2001 to May 2002, he worked as a student research associate with Logistics Management Institute, McLean, VA USA, in their Resource Analysis Group.

In 2002, he enrolled in the doctoral program in Industrial Engineering in the IE department (now known as Edward P. Fitts Department of Industrial and Systems Engineering) at NCSU, where he will earn a Ph.D. degree in August 2006. During the period May 2004–May 2006 he worked part-time as a technical student in the Operations Research group at SAS Institute, Cary, NC USA.

Acknowledgements

First and foremost, I thank Dr. Elmaghraby for his help, guidance, and support — financial, intellectual, and moral — during the course of my graduate study at NCSU. I consider myself very fortunate to have had the opportunity to work with him. I thank my committee members, Dr. Jim Wilson, Dr. Matt Stallmann, Dr. Ghosal, and Dr. Artiba for their insightful comments and for carefully reviewing my dissertation. I thank Dr. Lunardi for attending my preliminary and final exams as the graduate school representative. I am grateful to Dr. Fang and Dr. Nuttle for providing me with financial support at critical times during my graduate study. Thanks also to Cecilia for being such a wonderful graduate assistant.

I feel fortunate to have met a lot of nice people since arriving in Raleigh. Special thanks goes to Shashi Adiga, who made life a lot easier for me during the first few months. I will always appreciate his unflinching willingness to offer help and support. Thanks to Narahari for being such a fantastic person to live with. The last four years were memorable and I will definitely miss all the good times. Thanks to Abhinand and Shrini for their friendship and the wonderful times I spent hanging out with them. Thanks to all my friends — Harish, Gautam, Srivatsan, Bhavin, Srikant Nalatwad, Vinay, Rajasimhan, Raghuram, and many others — for making 'Apt. 2512-201' such a lively and entertaining place!

Thanks to my office mates Xiaoli, Sean, Yong, and others for their friendship and help. Special thanks to Burcu Özçam for being such a wonderful friend, and for all the help and support she provided during my stay at NCSU. Thanks also to Girish Kulkarni, Jawad, Sriram, Sridhar Dasu, Parikshit, and Akhil for being such good friends and peers — I enjoyed spending time with you guys!

Thanks to Naveen, Anjali, and Tanay for being like family to me — I always enjoyed your company (and the food, of course!) and I will definitely miss you folks. Thanks also

to Vijay and Vaidehi for their friendship and wonderful hospitality.

I am extremely grateful to SAS Institute, Cary NC USA, for providing me financial support through the last two years of my doctoral study. Special thanks to Radhika and Gehan for providing me the opportunity to work with them. I learned a lot during that time, and also made some good friends. Lindsey, Emily, Nilesh, Richard, Laura, Bengt, Balaji, and the rest of the SAS ORD group — thanks for the numerous enjoyable lunch-time (and office-time!) discussions.

I am extremely thankful to my parents for their constant support and encouragement. I salute their patience and their unconditional faith in my abilities over the years. I am also thankful to my cousins and their families for their support during my stay in the USA. They have always been there whenever I needed them. Lastly, I like to thank my fiancee, Smita, for her support and patience during the last six months of my doctoral study.

Table of Contents

List of Tables viii

List of Figures ix

1 Introduction 1
- 1.1 Background . . . 1
- 1.2 Underlying Hypothesis . . . 1
- 1.3 Problem Statement . . . 2
- 1.4 Scope and Objectives of Research . . . 3
- 1.5 Organization of the Dissertation . . . 3

2 Literature Review 5
- 2.1 Introduction and Classification of RCPSP . . . 5
- 2.2 Unimodal and Multimodal RCPSP . . . 6
- 2.3 Optimal Resource Allocation . . . 11

3 Optimal Resource Allocation – Deterministic Case 13
- 3.1 Nonlinear Programming Model . . . 14
  - 3.1.1 Unconstrained Activities . . . 14
  - 3.1.2 Individually Constrained Activities . . . 19
  - 3.1.3 Aggregate Resource Constraints . . . 22
  - 3.1.4 Limitations of the NLP Model . . . 28
- 3.2 Integer Programming Formulation . . . 29
- 3.3 Setup of Computational Experiments for the IP Formulation . . . 31
  - 3.3.1 Computational Results: Category 1 . . . 32
  - 3.3.2 Computational Results: Category 2 . . . 35
  - 3.3.3 Observations from Computational Experiments . . . 35

4 Optimal Resource Allocation – Stochastic Case 36
- 4.1 Problem Statement and Assumptions . . . 36
- 4.2 Policy Iteration–like Approach . . . 37
  - 4.2.1 Continuous-Time Markov Chain . . . 38

4.2.2 Phase Type Distribution . . . 39
4.2.3 Details of the Policy Iteration–like Procedure . . . 41
4.2.3.1 Illustrative example . . . 41
4.2.4 Limitations of the PI-like Approach . . . 47
4.3 Overview of Stochastic Programming . . . 48
4.3.1 Application of Stochastic Programming in Stochastic Project Scheduling 50
4.3.2 Inapplicability of Stochastic Programming to Our Problem . . . 51
4.4 Approach Via Simulation-Cum Optimization . . . 54
4.4.1 Variance Reduction Techniques . . . 54
4.4.1.1 Latin Hypercube Sampling (LHS) . . . 55
4.4.2 Numerical Example . . . 57

5 Conclusions and Future Research **63**
5.1 Main Conclusions of the Research . . . 63
5.2 Directions for Future Research . . . 65

A Generation of Test Networks **71**
A.1 Initial Framework . . . 72
A.2 The Generation of the Network . . . 72

B Listing of AMPL Code **74**
B.1 Nonlinear Programming Model . . . 74
B.2 Integer Programming Model . . . 77
B.2.1 AMPL model file . . . 77
B.2.2 AMPL data file . . . 78

List of Tables

2.1 Unimodal RCPSP Classification 7
2.2 Multimodal RCPSP Classification 8

3.1 Parameters of the Example Project. 18
3.2 Optimal solution—No Resource Constraints; Fixed Deadline. 20
3.3 Optimal Node Realization Times—No Resource Constraints. 20
3.4 Optimal Solution — Individual Resource Constraints. 21
3.5 Optimal Node Realization Times — Individual Resource Constraints. 21
3.6 Cutsets in the Example Project. 23
3.7 Optimal solution — Aggregate Resource Constraints. 27
3.8 Optimal Node Realization Times — Aggregate Resource Constraints. 27
3.9 Resource Allocation across the *udc*'s. 27

4.1 Values of Expected Total Cost for Fixed Allocations (x_1, x_2) and Varying x_3 . 45
4.2 Sample Output of LHS 56
4.3 Data for Example Network 3 57
4.4 Summary of Results for Project Completion Costs from Monte Carlo (MC) and LHS Runs 59
4.5 P-deciles for the Resource Allocation Vector (Uniform (LHS) Case) 59
4.6 Lower and Upper Bounds for W_j 62
4.7 P-deciles of Total Cost for Varying Values of R (R = 5, 6, 7, 8, ∞) 62

List of Figures

3.1 Tardiness Cost Function. 17
3.2 Example Network 1. 19
3.3 Project with *udc*'s marked. 24
3.4 Comparison of Computation Times: R = 5, n = 20, T = 20, 40, 60, 80. . . 33
3.5 Comparison of Computation Times: R = 5, n = 30, T = 20, 40, 60, 80. . . 33
3.6 Comparison of Computation Times: R = 7, n = 40, T = 20, 40, 60, 80. . . 34
3.7 Comparison of Computation Times: R = 7, n = 50, T = 40, 60, 80. 34
3.8 Comparison of Computation Times: T = 40, n = 40, R = 7, 8, 9, 10. . . . 35

4.1 Example Network 2 and its State Space 42
4.2 Surface Plot of Resource Allocations versus Total Cost 46
4.3 Example Network 3. 52
4.4 Hierarchy of Cutsets (not a complete tree). 53
4.5 Example Network 4 (AoA). 61

Chapter 1

Introduction

1.1 Background

An important issue that looms high in the management of real life projects is that of *risk* and *uncertainty*. Concerns about risk are everyday worries of project managers. They recognize the uncertainty in their estimates of resources, cost, and time. The issue is not about *recognition*, but about *measurement* (how does one measure the risk involved in a particular action), and about *coping* with uncertainty, especially in relation to resource allocation.

We are concerned with the concrete actions (decision making) relative to resource allocation to mitigate the deleterious effects of deviations, most of them unanticipated (known only in probability), in the progress of the project. In other words, suppose that the risk of a particular activity (or subset of activities) defaulting on its completion time (or cost estimate, or resource needs) is unacceptable, *what can be done* to reduce the risk, if not completely eliminate it?

1.2 Underlying Hypothesis

It is the thesis of this proposal that uncertainty resides in two domains. The first is "external" to the activity, such as the weather conditions, worker absenteeism, equipment failure, etc. The second is "internal" to the activity and resides in estimates of its *work content* (or "effort"). Traditionally the focus has been on the former aspect (external) with little attention to the latter (internal). And yet there are many projects, especially those with

appreciable research and development content, in which the internal factors play the dominant role. Most importantly, even those studies that concerned themselves with the issues of resource allocation are marked by strict adherence to *deterministic* estimates of the various parameters, ignoring (or skirting) the issue of uncertainty in the estimates of these parameters.

Uncertainty in the work content of an activity then is typically expressed by the manager in the form "it requires between *l* and *u* man-weeks" (where *l* is the lower bound and *u* is the upper bound), with or without knowledge of the probability distribution of the work content. In the face of such uncertainty in the work content the manager still has to *decide on the resources to be devoted to the activity*. The duration of the activity then becomes the *consequence* of the resources allocated to the activity, not the *source* of the uncertainty.

This perspective changes the view of risk management in a radical fashion because now the decision is concerned with the *optimal resource allocation* (with its concomitant cost) in order to achieve the desired objective; namely complete the project within the prescribed due date (in order to avoid any penalty of tardiness) and with minimal cost of resources.

We do not suggest that one ignore the external factors; indeed, they must be taken into consideration, sooner or later. But we wish to focus on the internal factors because the uncertainty stemming from them is the form of uncertainty that can be managed *via* proper allocation of resources *dynamically*.

Thus our research has two pivotal elements that distinguish it from prior studies in this field. The first is that we consider uncertainty explicitly. And the second is that our take-off point is the *work content* of the activity, not its duration.

1.3 Problem Statement

We are given a project network of n activities defined by a graph $G = (N, A)$, in which *A* is the set of arcs defining the activities and the precedence relations among them (AoA mode of representation). Let $w_j(r)$ denote the *work content* of activity j relative to resource r and $x_j(r)$ be the allocation of resource r to activity j. Define c_r to be the per-unit cost of usage of resource r, and c_t to be the cost of tardiness per unit time of delay. Furthermore, let T_s

be the project due date, so that if the project completes at time t_m[1], then the total cost of tardiness would be $c_t \cdot \max\{0, t_m - T_s\}$. Our objective is to optimally allocate resources to the activities such that the overall cost of resource allocation and tardiness is minimized.

1.4 Scope and Objectives of Research

The primary objective of this research is to address the problem of optimally allocating resources, that are limited in availability, to a project while minimizing a certain economic objective. the specific objectives can be summarized as follows:

1. Develop a mathematical model to solve the deterministic resource allocation problem when there are (a) unlimited resources, (b) resource availability constraints on individual resources, and (c) additional aggregate resource availability constraints across activities that can be performed in parallel.

2. Develop mathematical models that can address the stochastic version of the problem. Ideally, models that can be easily extended from the deterministic version.

3. Validate the mathematical models for correctness and robustness.

1.5 Organization of the Dissertation

The rest of this dissertation is organized as follows. The first part of Chapter 2 is a brief review of literature that describes the Resource-Constrained Project Scheduling Problem (RCPSP) and its variants, followed by some discussion on different methods used to solve them. It also draws attention to certain classes of problems that are similar to the problem addressed by our research. Chapter 3 discusses in detail the deterministic version of the resource allocation problem together with the two mathematical formulations — a non-linear programming (NLP) approach, based on the concept of uniformly directed cutsets (*udc*), and an integer programming (IP) model that is time-based. In Chapter 4 we address the stochastic version of the problem. We first discuss the Continuous-Time Markov Chain

[1] If the project completion time is a random variable, then we denote it by Υ.

(CTMC) approach when the work content of all activities are independent exponential random variables. We briefly outline the solution strategy and discuss the limitations of the approach in practice. The latter half of Chapter 4 is devoted to the case where the work content of activities follow a general probability distribution (i.e., not necessarily exponential). The limitations of standard stochastic programming approach are discussed followed by details of the alternate approach of simulation-cum optimization. Efficient sampling methods to reduce sample size and sample variance are also described. Finally conclusions and recommendations for future research are outlined in Chapter 5.

Appendix A describes the methodology used in generating the test cases for applying and validating the proposed mathematical models. Appendix B includes (i) AMPL model file used to solve the example in Chapter 3, and (ii) AMPL model and data files that illustrate the IP formulation.

Chapter 2

Literature Review

This review is organized into three sections. In Section 2.1 we describe the Resource-Constrained Project Scheduling Problem (RCPSP) and its classification. In Section 2.2 we survey the work done on the *unimodal* and *multimodal* RCPSP. We are particularly interested in the *multimodal* problem mainly because it relates closely the concept of *work content* advocated here. We provide a review of the work done in the the variants of the RCPSP, including a brief discussion of the stochastic instance of the problem. Finally in Section 2.3 we give an outline of the recent work done on the problem of optimal resource allocation in activity networks (Tereso et al. [2001, 2003b]).

2.1 Introduction and Classification of RCPSP

Resource-constrained Project Scheduling Problems (RCPSPs) involve scheduling the activities to satisfy constraints on the availability of resources or with limited capacity in order to meet some predefined objective related to the due date. As we shall see there are many different objectives, and these depend on the goals of the decision maker. The most common objective seems to be minimization of makespan, i.e., minimum time to complete the entire project. We shall discuss several variants of the RCPCP, beginning with the *unimodal* one and moving on to the *multimodal* problem. *Unimodal*, or single-mode problems, imply that each activity has a single execution mode. In other words both the activity duration and its requirement for a set of resources are fixed and known in advance. In *multimodal* problems, on the other hand, each activity can be processed in one of several modes (levels

of resource application, or several “substitutable" resources). Each mode implies a different option in terms of cost and/or duration. We broadly classify the RCPSP into the following six categories:

1. *Unimodal* RCPSP (SM–RCPSP)
2. *Multimodal* RCPSP (MM–RCPSP)
3. RCPSP with nonregular objective functions
4. Stochastic RCPSP
5. Multi–resource constrained Project Scheduling Problem (MRCPSP)

We focus on the first four items in our review. See Garey et al. [1976] for more on the relationship between RCPSP and the bin packing problem.

2.2 Unimodal and Multimodal RCPSP

Unimodal RCPSPs are classified into four general categories Herroelen et al. [1998a]. Throughout our discussion we shall assume that a project is represented as a graph $G = (N, A)$ where the arcs represent the activities and nodes represent the start or completion of the activities (AoA mode of representation). When we refer to precedence relations we consider two main types—one where an activity can start any time after the completion of its predecessor(s), and the other where the successor(s) can start only after the passage of some specified time after the finish of the activity. We denote the former type by ‘S’ (for simple succession) and the latter by ‘FSL’ (for finish-to-start lag). We also classify the problems based on the resource availability and requirement. Resource availability may be the same for all periods, or it may vary from period to period. Resource requirements for an activity may also be constant throughout its duration, or it may vary over time. We distinguish problems based on whether preemption of activities is allowed or not. Finally we classify problems based on the objective function. As mentioned earlier, the most common objective is to minimize makespan. However, there are other objectives that we refer to as ‘Regular,’ such as:

1. minimization of weighted activity tardiness.
2. minimization of the total number of tardy activities.
3. minimization of weighted mean flow time.

Table 2.1 summarizes our classification of *unimodal* RCPSPs. The Basic SM-RCPSP has been studied extensively in the literature. Typical solution strategies for this class of problems are through mathematical programming techniques: integer programming and LP relaxation, combined with implicit enumeration (branch-and-bound), and dynamic programming.

Table 2.1: Unimodal RCPSP Classification

	Basic SM-RCPSP	**SM-RCPSP with Preemption**	**General Precedence SM-RCPSP**	**General RCPSP**
Objective	Min makespan	Min makespan	Min makespan	Regular
Precedence	S	S	S	FSL
Resource availability per period	Constant	Constant	Constant or Time-varying	Constant or Time-varying
Resource requirement per period	Constant	Constant	Constant or Time-varying	Constant or Time-varying
Preemption	No	Yes	No	No

A variety of branch-and-bound algorithms have been developed for this problem. Most use partial feasible schedules as a starting point and then extend them in the branching process until a complete schedule is found. Owing to the computational complexity of the RCPSP, several priority-rule-based heuristics have been proposed. For more details on the various branching and pruning strategies and heuristic approaches see Herroelen et al. [1998a].

Resources can be classified as *renewable* or *non-renewable*. Non-renewable resources get depleted with consumption while renewable resources have the same amount of availability

Table 2.2: Multimodal RCPSP Classification

	Non-renewable resource MM-RCPSP	**Renewable resource MM-RCPSP**	**General MM-RCPSP**
Objective	Min makespan	Min makespan	Regular
Precedence	S	S	FSL
Resource availability per period	Constant within mode	Constant within mode	Constant within mode
Resource requirement per period	Constant within mode	Constant within mode	Constant within mode
Preemption	No	No	No
Trade-off	Time/Resource	Time/Resource	Time/Resource Time/Cost Resource/Resource

throughout the duration of the project.

In the *multimodal* RCPSP problem, a set of allowable execution modes can be specified based on the estimation of *work content* of the activity. Each mode is characterized by a processing time and an amount of resource of a particular type needed for completion of the activity. A summary of classification of the *multimodal* RCPSP problems is shown in Table 2.2. In the non-renewable–resource RCPSP, tasks are associated with non-renewable resources. A cost of resource r while processing task i, $c_i(r)$, is associated with this mode. If the set of modes for every task can be represented as a closed interval and $c_i(r)$ is an affine and decreasing function of processing time p_i, we have a linear time-cost trade-off problem. If the set has discrete values of $c_i(r)$ then we have a *discrete time-cost trade-off* (DTCT) problem.

There are two related problems — the *budget problem* (minimize makespan subject to budget constraints), and the *due-date problem* (minimize total cost, given a due date for each task). For the linear time-cost trade-off, Fulkerson [1961] developed an algorithm

that iteratively calculates a sequence of cuts in the AoA (Activity-on-Arc) network to compute the project cost curve. For the discrete case, the best known algorithms still rely on dynamic programming (DP), but additionally exploit the decomposition structure of the underlying network. All the reported algorithms for solving the DTCT problem exhibit exponential worst-case complexity. A notable exception is the dynamic program by Hindelang and Muth [1997], which claimed to execute in pseudo-polynomial time. However De et al. [1997] demonstrate that the DP is flawed, and offer a simple correction. They further prove that the DTCT decision problem is strongly $\mathcal{NP}$–complete by reduction from the 3–SAT problem (see Garey and Johnson [1979]). Based on this result they imply that the DTCT optimization problem is strongly $\mathcal{NP}$–hard. As mentioned earlier, this class of *multimodal* RCPSP relates very closely to the problem of concern to us in this thesis, and we shall revisit it in Chapter 3.

The renewable-resource RCPSP has been studied in some detail with respect to makespan minimization objective. Demeulemeester et al. [1997] present a branch-and-bound procedure based on the concept of maximal acivity-mode combination. An activity-mode combination is a subset of activities executed in a specific mode; it is maximal if no other activity can be added without causing a resource conflict. Sprecher and Drexl [1998] developed a branch-and-bound procedure that relies on an enumeration scheme based on the “precedence tree" concept. Several heuristics have been developed as well; the recent genetic algorithm (GA) approach by Hartmann [1998] being worthy of note.

In the stochastic RCPSP, the processing time of any activity is a random variable (r.v.) that follows some probability distribution. This problem class, while more realistic, leads to much greater complexity in analysis. The commonly pursued objective is the minimization of the *expected* makespan. Since a project contains many interdependent activities, thereby leading to interdependent activity completion times, the probability distribution of the project completion time is extremely difficult to determine. Often it prompts the assumptions of independence for tractability of analysis. Such assumptions, however, can lead to extremely misleading results in practice.

The literature on the stochastic version of DTCT is virtually void. Wollmer [1985] discussed a stochastic version of the linear time/cost trade-off problem. He discussed extensions to the problem of minimizing the *expected* project completion time subject to a budget

constraint and the problem of achieving a feasible fixed *expected* project completion time at minimum cost as well as generation of the project curve. Gutjahr et al. [2000] describe a stochastic branch-and-bound procedure for solving a specific version of the stochastic DTCT where so-called *measures* (like the use of manpower, the assignment of skilled labor, etc.) may be used to increase the probability of meeting the project due date, thereby avoiding penalty costs. Stork [2001] provides an excellent insight into the stochastic resource scheduling problem. He addresses the objective of makespan minimization and provides branch-and-bound algorithms. He uses the classical *critical path* lower bound and implements some clever sorting rules, preselective policies, and dominance rules to prune the search tree. See the handbook by Demeulemeester et al. [1997] for more details.

In a recent publication, Golenko-Ginzburg and Gonik [1997] developed a heuristic procedure for the RCPSP with stochastic activity times, with the objective of minimizing expected project duration. The procedure operates in stages where a decision is made to schedule the next activity based on the precedence constraints and current resource availability. If several activities are fighting for limited resources, a multiple knapsack problem is solved to select the next activities to be scheduled. The objective function of the knapsack formulation represents the maximum total contribution of the selected activities towards the expected project duration. The individual contribution is calculated as the product of the average activity duration and the probability of being in the critical path during the course of the project. The probability values are approximated by frequency values found via simulation. The rationale for the design is to give priorities to critical activities because of their significant impact on the expected project duration. Another recent publication dealing with the RCPSP with stochastic activity times is due to Fernandez and Armacost [1996]. This article warns users of off-the-shelf scheduling software of the danger of omitting constraints that are nonanticipatory. Specifically, they mention that Monte Carlo based approaches, such as those in Opera® and @RISK for Project, do not incorporate nonanticipatory constraints and therefore the output may be misleading. The primary output for these software packages generally is the empirical distribution of project duration. Since the procedures in these packages solve each scenario separately without enforcing nonanticipatory constraints, they arrive to non-implementable solutions (i.e., solutions that are based on information that is not available to the decision-maker at the time the decision

was made). The work of Valls et al. [1998] provides a good example of application of stochastic programming to stochastic project scheduling under resource constraints. They deal with the problem of resource-constrained project scheduling with stochastic activity interruptions, where they minimize the total expected value of weighted tardiness. We discuss their problem in detail in section 4.3.1 of Chapter 4.

2.3 Optimal Resource Allocation

The literature has been of little help on the problem of resource allocation in project networks. Tereso et al. [2001, 2003a,b, 2004] deal with resource allocation in multimodal activity networks. They address the stochastic version of the problem and assume exponential distribution for the *work content* of activities.

In the first paper (Tereso et al. [2001]), they present a dynamic programming approach, in which the definition of the *stage* depends on a special subset of activities that are to be "conditioned." We further elaborate their algorithm as follows. Consider a directed acyclic graph, $G = (N, A)$, where the set A denotes the set of activities. The special set of activities (denote it by $\mathcal{F}$) comprises activities for which the resource allocation is "fixed" — that is, they are no longer decision variables. The *s*tage of the dynamic program (DP) is defined as the epoch of decision on the value of x_j for an activity j in the set of decision variables, denoted by $\mathcal{D}$, and defined as $\mathcal{D} \equiv A - \mathcal{F}$. At each stage of the DP iteration, one decision variable is optimized. Thus there are as many stages as the size of the set $\mathcal{D}$. The *state* of the DP is defined as the vector of realization times of a subset of nodes that enable the decision $x_j, j \in \mathcal{D}$ to be made and the stage "reward" be evaluated. The stages are numbered backwards, so 'stage k' means 'k stages to go' to complete the project. Let $x_{[k]}$ denote the decision variable in stage k, for $k = 1, \ldots, K$. Further, 'stage 1' is the stage containing the terminal node m and has the decision variable $x_{[1]}$. The state in stage k is denoted by $s_{[k]}$. For all stages except stage 1, the stage reward (denoted by $C_{[k]}(x_{[k]}, s_k)$) is simply the resource cost, a random variable, defined by $x_{[k]}W_{[k]}$, where $W_{[k]}$ refers to the work content in stage k. In stage 1, the stage reward is the sum of the resource cost and the tardiness cost, if any; $(= c_t \cdot \max\{\, 0, t_m - T_s \,\})$, where c_t is the tardiness cost per unit time delay, t_m is the time of realization of the terminal node, and T_s is the due date. Let

$f_k(s_k|\,\mathcal{F})$ denote the minimal cost at stage k when the state is s_k, and conditioned on the allocation in $\mathcal{F}$. The DP functional equation is defined as

$$f_k(s_k|\mathcal{F}) = \min_{x_{[k]}\in\mathcal{D}} \mathbb{E}\left\{C_{[k]}(x_{[k]}, s_k) + \mathbb{E}\left[f_{k-1}(S_{k-1}|\,\mathcal{F})\right]\right\},$$

where S_{k-1}, a $r.v.$, represents the realization time of state s_{k-1}, $k = 2, \ldots, K$.

The optimum is secured by removing the conditioning, and the final solution is achieved by evaluating the following:

$$f(s_K = 0) = \min_{\mathcal{F}} f_K(s_K|\,\mathcal{F}).$$

The solution obtained via dynamic programming yields a policy that prescribes the optimal resource allocation under every conceivable state of the project as it progresses over time.

The DP model just discussed is computationally very intensive. Just to get an idea, if there are q different allocation values of an activity in set $\mathcal{F}$, and accordingly $q^{|\mathcal{F}|}$ possible allocations from which the optimum is selected. Tereso et al. [2003b] present some basic approximations to the DP model such as (i) replacing the work content of activities in the set $\mathcal{F}$ with their respective mean values, (ii) replacing the work content of all activities by their respective mean values, and (iii) using a nonlinear programming (NLP) model. They also report the performance of the stochastic global optimization technique, *Electromagnetism-like Approach* of Birbil and Fang [2003], when applied to their problem.

Chapter 3

Optimal Resource Allocation – Deterministic Case

In this chapter we discuss the scenario where the work content of each activity is deterministic. We analyze three possible cases, namely

1. Unconstrained activities — Unlimited resources available
2. Individual resource constraints on activities — Resource usage for each activity is constrained between upper and lower bounds.
3. Aggregate resource constraints — In addition to individual resource constraints, the resource usage across different activities running in parallel at the same time is constrained between upper and lower bounds. Resource(s) availability may be constant over time, or it may vary from period to period.

In the first part of the chapter we develop a nonlinear programming (NLP) model as we progress from the simple unconstrained case to the more complex aggregate-constrained case. We use an example project as an aid to explain the various features and intricacies of the model. We then explain the limitations of the NLP model and conclude the discussion with the presentation of some computational results. Next we develop an integer programming (IP) formulation to address the limitation of the NLP model. The need for two mathematical models is to be able to handle continuous and discrete allocation of resources. The NLP model handles the former and the IP model, the latter. At the end of the chapter we present computational results of the IP model and comment on its performance.

Before we proceed with the mathematical program formulation, we wish to clarify some notation that will be often used. We use the term *path* to refer to a set of activities in series

that lead from one node to another. We define the terms *cutset* and *uniformly directed cutset (udc)* as follows:

Assume we have a graph $G = (N, A)$, where N is the set of nodes and A is the set of arcs defining the activities and the precedence relations among them. Let $B \subset N$ and $\bar{B} = N - B$. Let i and i' denote the start and end nodes of activity j, respectively. Furthermore, define $\left(B, \bar{B}\right) = \left\{j \in A : i \in B, i' \in \bar{B}\right\}$.

Definition 3.1. *The set of edges $\left(B, \bar{B}\right)$ is called an (s,t)-cut (or cutset) if $s \in B$ and $t \in \bar{B}$.*

Definition 3.2. *An (s,t)-cut $S \equiv \left(B, \bar{B}\right)$ is called a uniformly directed cutset (udc) if $\left(\bar{B}, B\right)$ is empty. In other words, an (s,t)-cut is a udc if and only if no two arcs in the cut belong to the same path in the network.*

3.1 Nonlinear Programming Model

We consider a very general model with more than one type of renewable resource. We incorporate general (nonlinear) functions for activity duration, resource usage cost and tardiness cost. For the sake of convenience we state the notation used as follows:

$w_j(r)$	:	work content of activity j relative to resource type r; work content is typically expressed in units such as man-hours, man-days, machine-hours, etc.
$x_j(r)$	:	amount of resource type r (men, machines, etc.) allocated to activity j
$y_j(r)$	:	duration of activity j corresponding to allocation $x_j(r)$
$C_j(R)$	:	total cost of resource usage for activity j
C_R	:	total cost of resource usage for all activities
C_T	:	total cost of tardiness

Note: If only one type of renewable resource is available, one can simply use the notations w_j, x_j, and y_j to refer to work content, resource allocation, and activity duration, respectively.

3.1.1 Unconstrained Activities

We start with the simplest scenario. There is more than one critical resource, and for the sake of illustration we shall assume that there are two, referred to generically as *r*, both

of abundant availability, at a price, of course. The duration of the activity, $y_j(r)$ can be modeled as a function of the resources allocated to it as

$$y_j(r) = \frac{w_j(r)}{x_j^{\gamma}(r)}, \tag{3.1}$$

where the exponent γ is constrained to be $\gamma \in [0.5, 1.0]$. For instance, if $\gamma = 0.5$, then doubling the resource allocation will decrease the duration by only about 30%,

$$\frac{y_2}{y_1} = \frac{\left(w/\sqrt{2x}\right)}{\left(w/\sqrt{x}\right)} = \frac{1}{\sqrt{2}} = 0.707.$$

On the other hand, if $\gamma = 0.75$, then

$$\frac{y_2}{y_1} = \frac{1}{2^{0.75}} = \frac{1}{1.68} = 0.59,$$

which means that the duration is reduced by about 41%, etc. Typically one would encounter such functional relationship between resource allocation and resulting duration in software development projects. Observe that if $\gamma > 1$ (representing synergism) then one can achieve super-linear improvement in duration by increased resource allocation.

The duration of the activity, denoted by y_j, is the *maximum* of the individual durations secured from each resource,

$$y_j = \max\left\{y_j(r)\right\}_{r=1,2}.$$

Since it does not make sense to incur cost unnecessarily due to excessive resource allocation, it must be true that (taking note of (3.1)),

$$y_j(r_1) = \frac{w_j(r_1)}{x_j^{\gamma_1}(r_1)} = \frac{w_j(r_2)}{x_j^{\gamma_2}(r_2)} = y_j(r_2),$$

and hence,

$$x_j(r_2) = \left(\frac{w_j(r_2)}{w_j(r_1)}\right)^{\frac{1}{\gamma_2}} x_j^{\frac{\gamma_1}{\gamma_2}}(r_1). \tag{3.2}$$

One may perform a data pre-processing step to calculate the ratios

$$\left(\frac{w_j(r_2)}{w_j(r_1)}\right)^{\frac{1}{\gamma_2}} \quad \text{and} \quad \frac{\gamma_1}{\gamma_2},$$

which are denoted by ϕ_j and $\gamma_{1,2}$, respectively. In other words,

$$\phi_j = \left(\frac{w_j(r_2)}{w_j(r_1)}\right)^{\frac{1}{\gamma_2}} \tag{3.3}$$

$$\gamma_{1,2} = \frac{\gamma_1}{\gamma_2} \tag{3.4}$$

Henceforth one may speak of $x_j(r_2)$ in terms of $x_j(r_1)$ to yield the same duration,

$$x_j(r_2) = \phi_j x_j^{\gamma_{1,2}}(r_1). \tag{3.5}$$

Likewise, y_j can be expressed in terms of resource r_1 only,

$$y_j = w_j(r_1) \cdot x_j^{-\gamma_1}(r_1). \tag{3.6}$$

The activity's cost of resources usage, $C_{j,R}$ assumes the following form:

$$C_j(R) = \sum_{r=1,2} k_j(r)\, x_j^2(r) \cdot y_j \tag{3.7}$$

in which $k_j(r)$ is a constant of proportionality which may vary with the activity j and the resource r. Note that we also assume the resource usage cost (for both types of resources) to be quadratic in the duration of the activity, hence the exponent 2 for $x_j(r)$. It is possible that it could vary with resource r and activity j. Substituting for y_j from (3.6) and for $x_j(r_2)$ from (3.5) into (3.7) we get,

$$\begin{aligned} C_j(R) &= k_j(r_1) \cdot x_j^2(r_1) \cdot y_j + k_j(r_2) \cdot x_j^2(r_2) \cdot y_j \\ &= y_j \left[k_j(r_1) \cdot x_j^2(r_1) + k_j(r_2) \cdot \left(\phi_j x_j^{\gamma_{1,2}}(r_1) \right)^2 \right], \end{aligned} \tag{3.8}$$

a function of only $x_j(r_1)$.

The total cost of both resources (recall that we are assuming only two resources) is simply the sum of $C_j(R)$ over all the activities

$$C_R = \sum_{j \in A} C_j(R). \tag{3.9}$$

We are now ready to deal with the cost of tardiness. To this end we assume that the project has a specified due date T_s, and that the cost of tardiness is *piecewise linear* and convex in the tardiness[1], with slopes $p_1 < p_2 < \cdots$. (This can be used to closely approximate any convex nonlinear function of tardiness.) Assuming that the project is completed at time t_m, the total cost of tardiness is given by,

$$C_T = \begin{cases} p_1(t_m - T_s), & \text{if } T_s \le t_m \le b_1, \\ p_1(b_1 - T_s) + p_2(t_m - b_1), & \text{if } b_1 \le t_m \le b_2, \\ \text{etc.} \end{cases} \tag{3.10}$$

[1] An incentive for completing the project before the specified time T_s can be easily incorporated into the model. We forfeit such generalization for the sake of simplicity.

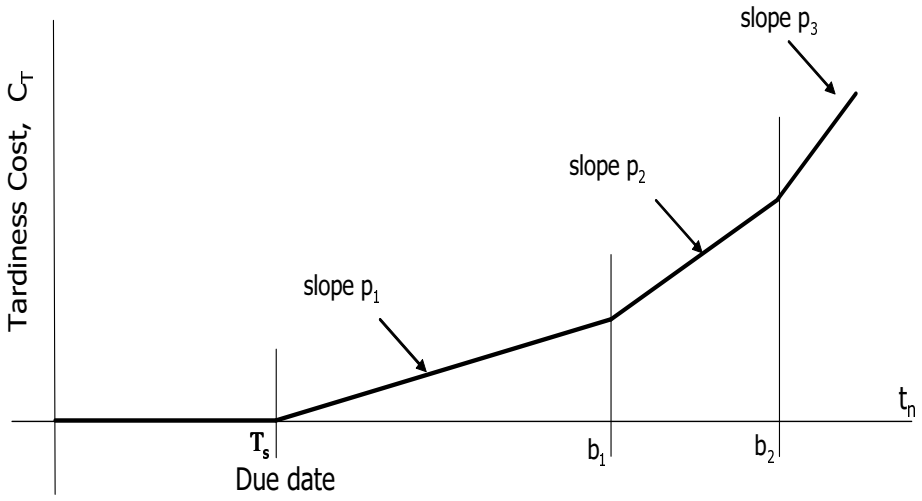

Figure 3.1: Tardiness Cost Function.

The costing of tardiness would appear as shown in Fig. 3.1.

Based on these assumptions it is easy to see that the objective is to determine the allocation vector X^* that achieves:

$$\min_X \; z\left(X\right) = C_R + C_T \tag{3.11}$$

The only “structural" constraints we have to deal with are the precedence constraints. Let t_i denote the time of realization of node *i*, with the assumption that $t_1 = 0$ (the project starts at time 0) and the nodes in the AoA representation of the project are numbered topologically so that an arrow always leads from a smaller number to a larger one. Then the realization of node *m* signals the completion of the project. We have,

$$t_{i'} \geq t_i + y_j, \;\; \forall j \equiv \left(i, i'\right) \in A. \tag{3.12}$$

Total tardiness is defined as $v = \max\left\{0, \left(t_m - T_s\right)\right\}$, which is replaced by the constraint

$$v \geq t_m - T_s, \tag{3.13}$$

together with the requirement that $v \geq 0$.

Finally we have the nonnegativity constraints,

$$x_j \geq 0, \; \forall \, j \in A. \tag{3.14}$$

We now consider the "logical" constraints imposed by the nature of the problem, which stem from the desire to equate the durations of paths in parallel. Consider an example network shown in Figure 3.2. We have,

$$\text{Activities 4 \& 5 are in parallel :} \quad y_4 = y_5 \tag{3.15}$$

$$\text{Activities 6 \& 7 are in parallel :} \quad y_6 = y_7 \tag{3.16}$$

$$\text{2 parallel paths}^{\dagger} \text{ from node 2 to node 5 :} \quad y_2 + y_4 = y_3 + y_6 \tag{3.17}$$

$$\text{Activities 8 \& 9 are in parallel :} \quad y_8 = y_9 \tag{3.18}$$

† A path is identified by the activities on it.

The mathematical program defined by the objective of Equation (3.11) with its variables defined in Equations (3.5)–(3.10), and constraints (3.12)–(3.18) is the desired model for this problem.

Consider the project network shown in Figure 3.2 with values of the various parameters as given in Table 3.1. The *work content* is measured in man-days. The due date $T_s = 36$, with tardiness cost cut-off points $b_1 = 5,\ b_2 = \infty$, and slopes $p_1 = 200,\ p_2 = 800$. The interpretation of the pair ($b_2 = \infty$, p_2 = 800) is that beyond the cut-off point b_1 = 5, the tardiness is penalized at the rate of 800. The resource usage coefficients (exponents for $x_j(r)$ in (3.7)) are 1.5 for resource 1 and 2 for resource 2. The resource coefficients are $\gamma_j(1) = 0.90,\ \gamma_j(2) = 0.95$, the same for all activities. The resource cost coefficients are also the same for all activities, with $k_j(1) = 6,\ k_j(2) = 8$.

Table 3.1: Parameters of the Example Project.

Activity	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
$w_j(1)$	25	31	38	16	23	16	15	29	8	14
$w_j(2)$	34	42	17	24	26	21	11	33	12	18

The optimal solution (obtained using AMPL/MINLP-BB[2] on NEOS[3] optimization server) to this set of parameters is as shown in Table 3.2. The corresponding node realization times

[2]See Fletcher and Leyffer [1998] and Leyffer [2001] for details on the MINLP-BB algorithm.

[3]See Czyzyk et al. [1998], Gropp and Moré [1997], Dolan [2001] for details on the NEOS Optimization Server.

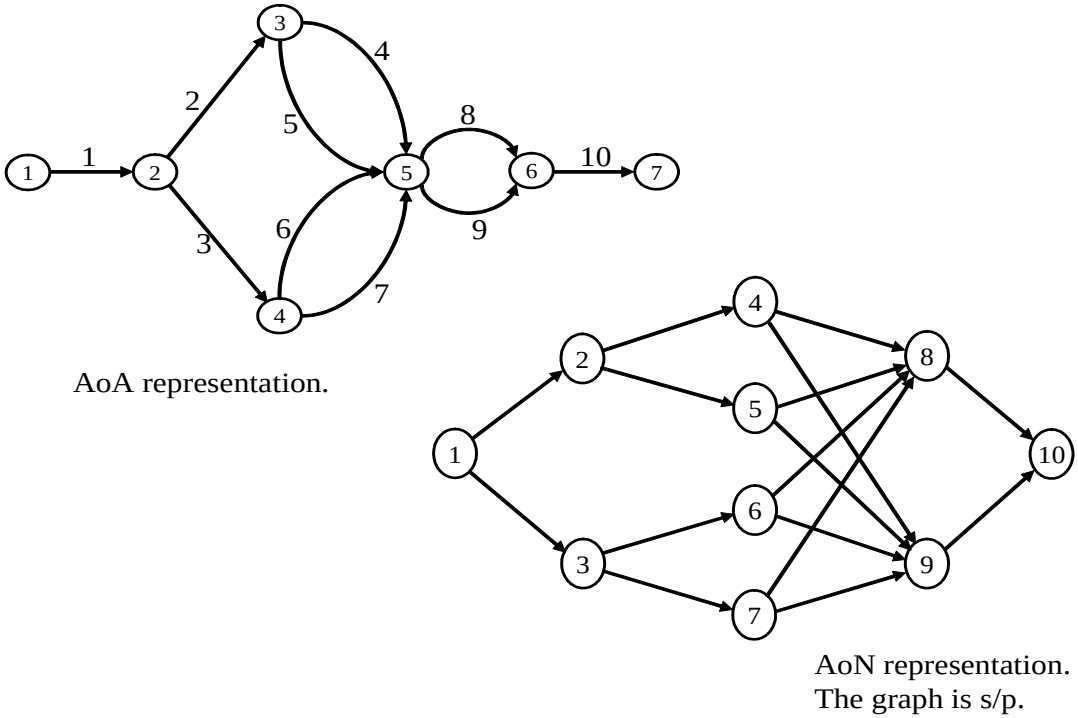

Figure 3.2: Example Network 1.

are shown in Table 3.3. Observe that the optimal solution reached, but did not exceed, the cut-off value of $b_1 = 5$, mainly due to the very high cost of tardiness beyond $b_1 = 5$.

3.1.2 Individually Constrained Activities

We continue to assume that we know deterministically the *work content* w_j for each activity in the project, that we have only two resources to contend with, and that the work content of activity j relative to the two resources are denoted by $w_j(r)$, $r = 1, 2$. The definition of the the variables remains as before, together with the costing relative to resource usage and tardiness.

But now we assume that the amount of each resource allocated to the activity is individually constrained,

$$0 < l_j(r) \leq x_j(r) \leq u_j(r) < \infty, \quad r = 1, 2 \tag{3.19}$$

Table 3.2: Optimal solution—No Resource Constraints; Fixed Deadline.

j	$x_j\,(1)$	$x_j\,(2)$	y_j	$C_j(R)$
1	3.64	4.69	7.82	1,705.14
2	3.18	3.98	11.30	1,856.92
3	4.05	1.61	10.80	751.82
4	1.74	2.59	9.71	656.11
5	2.61	2.82	9.71	863.24
6	1.65	2.14	10.21	502.45
7	1.53	1.08	10.21	211.92
8	4.18	4.44	8.00	1,674.72
9	1.00	1.53	8.00	198.28
10	3.84	4.67	4.17	914.09
			Resource Cost	$9,334.68$
			Tardiness Cost	1000.00
			Total Cost	$10,334.68$

Table 3.3: Optimal Node Realization Times—No Resource Constraints.

Node, i	1	2	3	4	5	6	7
Time, t_i	0	8	19	19	29	37	41

The rationale for such bounds may run as follows. The lower bound may represent the least amount of resource needed to "get the activity going," or the least amount of resource acquisition possible (*e.g.,* you cannot rent a truck for less than half a day), and the upper bound may represent the maximal allocation that the activity can bear, or the maximal availability of the resource in the marketplace, etc.

We now have two types of constraints: precedence constraints and resource constraints. The precedence constraints are the same as discussed earlier and shall not be repeated, see (3.12). The resource constraints are as defined in (3.19). Equation (3.13) defines tardiness. Finally, we have the nonnegativity constraints (3.14). The mathematical program is the same as in Section 3.1.1, but with the addition of constraints (3.19).

To exemplify, we have implemented the model on the same project network shown in

Table 3.4: Optimal Solution — Individual Resource Constraints.

j	$x_j(1)$	$x_j(2)$	y_j	$C_j(R)$
1	2.27	3.00	11.97	1,107.10
2	2.36	3.00	14.79	1414.05
3	2.96	1.20	14.29	602.38
4	1.86	2.76	9.16	696.27
5	2.78	3.00	9.16	914.22
6	1.75	2.27	9.65	530.92
7	1.63	1.15	9.65	222.35
8	2.76	3.00	11.62	1,156.81
9	1.00	1.53	11.62	198.28
10	2.41	3.00	6.34	598.84
			Resource Cost	$7,441.22$
			Tardiness Cost	$11,400.00$
			Total Cost	$18,841.22$

Table 3.5: Optimal Node Realization Times — Individual Resource Constraints.

Node, i	1	2	3	4	5	6	7
Time, t_i	0	12	27	27	36	48	54

Figure 3.2. The parameters are the same as in Table 3.1 with the following modification — each activity j has bounds $[l_j(r), u_j(r)]$ on the amount of resource r that can be allocated to it. For simplicity of exposition we assumed $[l_j(r), u_j(r)] = [1, 3]$ for all activities $j \in A$, and for both resources, $r = 1, 2$. The optimal solution to this set of parameters is shown in Table 3.4.

We make the following observations on this result — As expected, imposing resource allocation bounds on individual activities increased the overall project cost. The values for the bounds on resources were chosen in such a way that they would actually be *constraining*. This is evident from Table 3.4 where one can notice that there are indeed activities with resource allocations at the upper bound of three units. Notice that there has also been an increase in duration in each of the path segments, thereby pushing the project duration to

54 time units.

Whenever the resource allocation for an activity is between its bounds it corresponds to the optimal allocation either absolutely (relative to its resource cost) or conditional upon minimizing the tardiness cost. For instance, consider activity 1. Its resource cost is

$$\begin{aligned} C_1(R) &= y_1\left(6x_1^{1.5}(1)+8x_1^2(2)\right) \\ &= \left[6x_1^{1.5}+8\left(\left(w_1(2)/w_1(1)\right)^{(1/\gamma_2)}\times x_1^{\gamma_1/\gamma_2}(1)\right)^2\right]\times\left(\frac{w_1(1)}{x_1^{\gamma_1}(1)}\right) \\ &= 150x_1^{0.60}(1)+382.09x_1^{0.9947}(1), \end{aligned}$$

a function of only $x_1(1)$, which is minimized at $x_1(1)=1$, its lower bound. However, this would prolong the duration of activity 1 from its current value of 11.97 to ≈ 25. This, in turn, would prolong the completion time of the project by 13 units, assuming that other activities remain the same$(=25-11.97)$, costing an additional $13\times 800\approx 10,400$.

3.1.3 Aggregate Resource Constraints

We now address the most general model of this genre of problems in which, in addition to the bounds on individual resource usage for each activity, there is also an *aggregate* constraint on the total resource usage at any time. This constraint is present whenever the resource availability is limited. This scenario is akin, but not identical, to the RCPSP. The most important distinction is the concept of *work content* and the focus on obtaining optimal specification of the resource allocation to the activities. We submit that this is a more realistic scenario which reflects the managerial discretion in resource allocation. Further, our scenario can easily accommodate the classical RCPSP scenario by defining discrete values of possible resource allocations and its variants.

The variables, parameters, and constraints of the model in Section 3.1.2 remain intact in the model proposed here. But the introduction of the condition that at any time no more than the available capacity of each resource is utilized gives rise to the consideration of two aggregate capacities R_1 and R_2 (recall that we are still assuming that there are only two resources) so that the total resource allocated to any *concurrent* activities at any time *t* is at most R_r. This may be written generically as,

$$\sum_{concurrent} x_j(r) \le R_r, \quad r=1,2. \tag{3.20}$$

There remains the definition of “concurrent" activities, for which we must appeal to the concept of *uniformly directed cutset* (*udc*), which has a long and venerable history (see Herroelen et al. [1998a] for a summary review).

It is intuitively clear that at any time, progress in the project must be along activities that lie on a *udc*. The first reaction to this realization is to enumerate all *udc*'s and constrain the total resource usage in each one. But slight reflection reveals that such blanket coverage may result in an *over-constrained* model, mainly due to the possibility that an activity, or several activities, may have already completed their processing and they are no longer involved in the “current" *udc*. In which case, including the completed activities in the constraint for the new *udc* would then limit the resources available to the newly introduced activities, possibly resulting in a higher cost.

To fully comprehend the impact of such over-constraining, consider the example network of Figure 3.2, re-drawn in Figure 3.3 with all the *udc*'s displayed. In all, there are 7 *udc*'s, as shown in Table 3.6 below. We use the notation $S^{()}$ to denote a *udc*. When we use the prefix *cutset* with $S^{()}$, we are actually referring to the *udc*.

Table 3.6: Cutsets in the Example Project.

UDC	$S^{(1)}$	$S^{(2)}$	$S^{(3)}$	$S^{(4)}$	$S^{(5)}$	$S^{(6)}$	$S^{(7)}$
Activities	$\{1\}$	$\{2,3\}$	$\{2,6,7\}$	$\{3,4,5\}$	$\{4,5,6,7\}$	$\{8,9\}$	$\{10\}$

Suppose that progress is on-going on activities 2 and 3 in $S^{(2)}$ (referred to as activities 2 and 3 being “*active*"), and suppose activity 2 finishes first. Then activities 4 and 5 can be initiated and “control" moves to *cutset* $S^{(4)}$. To be sure, the total resource usage was first constrained by

$$\sum_{j \in S^{(2)}} x_j(r) \leq R_r, \quad r = 1, 2,$$

and then by

$$\sum_{j \in S^{(4)}} x_j(r) \leq R_r, \quad r = 1, 2.$$

When activity 3 is completed we can initiate activities 6 and 7, and “control" moves to cutset

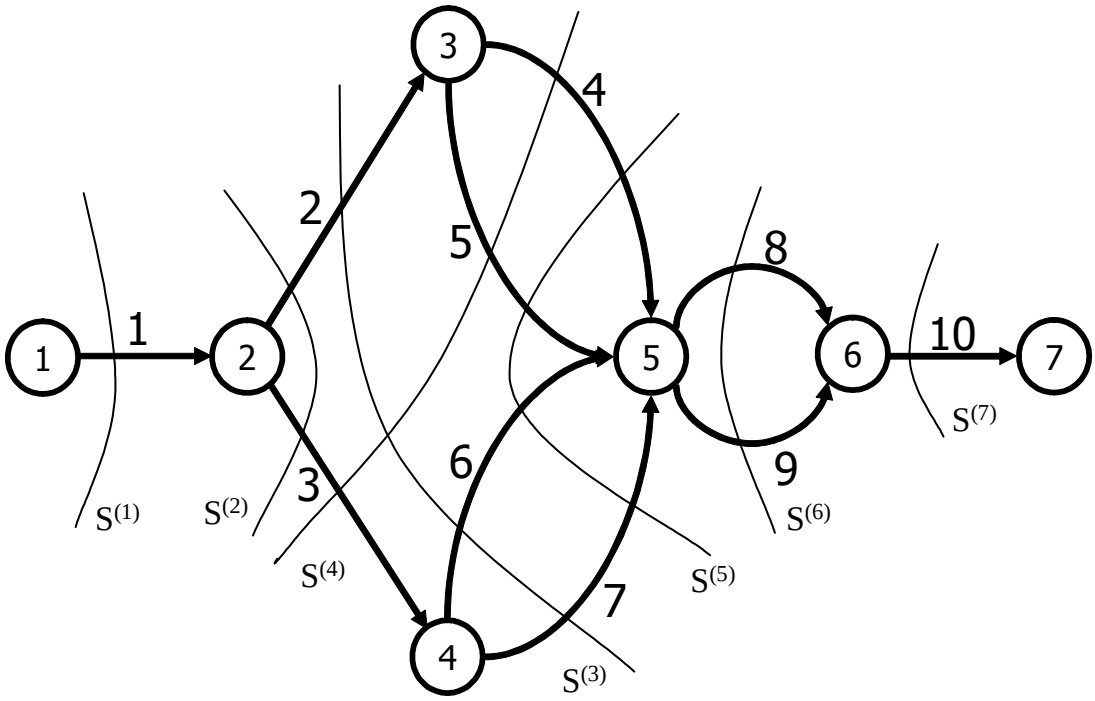

Figure 3.3: Project with *udc*'s marked.

$S^{(5)}$, which gives rise to the constraint,

$$\sum_{j \in S^{(5)}} x_j(r) \leq R_r, \quad r = 1, 2.$$

Observe that *cutset* $S^{(3)} = \{2, 6, 7\}$ never appeared, and its corresponding constraint

$$\sum_{j \in S^{(3)}} x_j(r) \leq R_r, \quad r = 1, 2$$

did not play any role in the definition of the resources allocated to any of the 6 activities 2,3,4,5,6,7! Taking such constraint into account would have "over-constrained" the allocation because the allocation to activity 2 may have been high, which would drastically confine the allocation to activities 6 and 7. A similar analysis can be made for the case in which activity 3 finishes first, or when both activities finish "almost at the same time."

In order to avoid such over-constraining the allocations, we proceed as follows. It is evident that either *cutset* $S^{(3)}$, $S^{(4)}$, or $S^{(5)}$ will come into existence after "control" escapes

cutset $S^{(2)}$, and that *cutset* $S^{(3)}$ will be the "controlling" *cutset* if activity 3 finishes first; *cutset* $S^{(4)}$ will be the "controlling" *cutset* if activity 2 finishes first; and *cutset* $S^{(5)}$ will be the "controlling" *cutset* if activities 2 and 3 finish "at the same time," or almost so. This translates into the following condition:

$$\begin{array}{lcl} \text{if } t_4 \leq t_3 - \varepsilon & : & S^{(2)} \rightarrow S^{(3)} \\ \text{else if, } t_3 \leq t_4 - \varepsilon & : & S^{(2)} \rightarrow S^{(4)} \\ \text{else, } |t_3 - t_4| < \varepsilon & : & S^{(2)} \rightarrow S^{(5)} \end{array} \tag{3.21}$$

In the above condition ε is some minimal spacing in time between the realization of the two nodes to be considered having been realized at different times.

This logic translates into the following set of constraints in which $\delta_{(\cdot)}$ and $\rho_{(\cdot)}$ are binary 0,1 variables, and M is a large number.

$$t_3 - t_4 \leq -\varepsilon + M\delta_{(3,4)} \tag{3.22}$$

$$t_4 - t_3 \leq -\varepsilon + M\rho_{(3,4)} \tag{3.23}$$

$$t_3 - t_4 \leq -\varepsilon + M\left(1 - \delta_{(3,4)} + \rho_{(3,4)}\right) \tag{3.24}$$

$$t_4 - t_3 \leq -\varepsilon + M\left(1 + \delta_{(3,4)} - \rho_{(3,4)}\right) \tag{3.25}$$

$$t_3 - t_4 \leq \varepsilon + M\left(2 - \delta_{(3,4)} - \rho_{(3,4)}\right) \tag{3.26}$$

$$t_4 - t_3 \leq \varepsilon + M\left(2 - \delta_{(3,4)} - \rho_{(3,4)}\right) \tag{3.27}$$

$$\sum_{j=3,4,5} x_j(r) \leq R_r + M\left(1 - \delta_{(3,4)} + \rho_{(3,4)}\right), \quad r = 1,2 \tag{3.28}$$

$$\sum_{j=2,6,7} x_j(r) \leq R_r + M\left(1 + \delta_{(3,4)} - \rho_{(3,4)}\right), \quad r = 1,2 \tag{3.29}$$

$$\sum_{j=4,5,6,7} x_j(r) \leq R_r + M\left(2 - \delta_{(3,4)} - \rho_{(3,4)}\right), \quad r = 1,2 \tag{3.30}$$

That this set of constraints represents the logic in (3.21) can be ascertained from the following tabulation:

$\delta_{(3,4)}$	$\rho_{(3,4)}$	**Consequences**
0	0	cannot be realized
0	1	(3.25), (3.29) are constraining
1	0	(3.24), (3.28) are constraining
1	1	either (3.26) or (3.27), and (3.30) are constraining

The reason why the (0,0) eventuality cannot be realized is that having both variables equal to 0 would cause inequality (3.22) to impose the constraint

$$t_3 \leq t_4 - \varepsilon$$

which would contradict inequality (3.24), which says that

$$t_3 \geq t_4 + \varepsilon.$$

Therefore one of the three conditions specified in (3.21) must be realized. The condition $t_3 < t_4 - \varepsilon$ is satisfied by the pair $(1, 0)$ and constraint (3.24), and the condition $t_4 < t_3 - \varepsilon$ is satisfied by the pair $(0, 1)$ and constraint (3.25); finally the condition $|t_3 - t_4| < \varepsilon$ is satisfied by the pair $(1, 1)$ and constraints (3.26) and (3.27). The respective resource constraints of (3.28)–(3.30) perform the desired restrictions on the resource allocation accordingly.

The above analysis reveals the difficulty in the implementation of this model. A set of nine constraints similar to (3.22)–(3.30), involving the definition of two *binary* variables, must be written for every pair of parallel nodes. The illustrative example of Figure 3.2 was selected advisedly because it manifests all the characteristics of larger projects and yet is small enough that it can be solved to optimality.

For illustrative purposes the model was solved to optimality assuming total availability of $R_1 = 8$, $R_2 = 8$, and $\varepsilon = 0.50$. The aggregate resource constraints, R_1 and R_2 were chosen in such a way that at least one of the constraints (3.20) is binding. Table 3.7 has the solution.

Activity 2 actually finishes at $t_3 = 26.76$, and activity 3 finishes at $t_4 = 26.26$; both are rounded off to 27 in Table 3.8. Since we assumed the minimal spacing $\varepsilon = 0.5$ for which the activities are considered to have completed at different times, and since the difference $|t_3 - t_4| = 0.5$ exactly, we consider that the two activities have completed at the same

Table 3.7: Optimal solution — Aggregate Resource Constraints.

j	$x_j\,(1)$	$x_j\,(2)$	y_j	$C_j(R)$
1	2.27	3.00	11.97	1,107.10
2	2.36	3.00	14.79	1,414.05
3	2.96	1.20	14.29	602.38
4	1.61	2.40	10.45	608.84
5	2.40	2.61	10.45	803.15
6	1.52	1.98	10.95	468.69
7	1.41	1.00	10.95	199.46
8	2.76	3.00	11.62	1,156.81
9	1.00	1.53	11.62	198.28
10	2.41	3.00	6.34	598.84
			Resource Cost	$7,157.60$
			Tardiness Cost	$13,000.00$
			Total Cost	$20,157.60$

Table 3.8: Optimal Node Realization Times — Aggregate Resource Constraints.

Node, i	1	2	3	4	5	6	7
Time, t_i	0	12	27	27	38	49	56

Table 3.9: Resource Allocation across the *udc*'s.

UDC	Resource 1	Resource 2
$S^{(2)}$	5.32	4.2
$S^{(3)}$	6.73	5.2
$S^{(4)}$	6.97	6.21
$S^{(5)}$	6.94	7.99
$S^{(6)}$	3.76	4.53

time, for all practical purposes. This is verified by the fact that both $\delta_{(3,4)}$ and $\rho_{(3,4)}$ are equal to 1. Therefore the controlling *cutset* (see Fig. 3.3) is $S^{(5)} = \{4, 5, 6, 7\}$, and the constraining equations are (3.26), (3.27), and (3.30). The above table indicates that the total requirements for this *cutset* of resource 1 is ≈ 6.94 and for resource 2 is ≈ 7.99. This is in line with our assertion earlier.

3.1.4 Limitations of the NLP Model

As seen in the case of aggregate constraints, one needs to identify nodes that are in "parallel" and then impose a set of 9 constraints similar to (3.22)–(3.30) for each pair. But in general, if there are p paths in parallel, each containing k nodes, there shall be $9k\binom{p}{2} = 9kp\,(p-1)\,/2$ such constraints involving the definition of $kp\,(p-1)$ integer-valued variables. Of course this is in addition to the constraints of the model specified in Section 3.1.2, some of which are nonlinear.

Another major drawback of the NLP model is the additional complexity that arises when the network is not series/parallel. In order to be able to apply the set of constraints (3.22)–(3.30) one needs to first "condition" on some arcs (i.e., activities) and then proceed with optimization. These arcs later need to be "unconditioned" in order to reflect the original scenario, and this process is of exponential complexity.

3.2 Integer Programming Formulation

The biggest drawback of the NLP formulation, as alluded to earlier, is that it cannot be used when the underlying project network is not series/parallel. We therefore shift our focus to "time interval-based" monitoring of resources as opposed to "cutset-based" approach in the NLP model. In other words, we have an integer program which incorporates discrete-time indexing of variables that correspond to resource usage.

We now restate our problem in the context of a single-type renewable resource availability. We are given a project $G = (N, A)$, where $N = (1, 2, \ldots, m)$ nodes and $A = (1, 2, \ldots, n)$ activities. Each activity $j \in A$ has a work content w_j. Denote by x_k, the amount of resource in mode k allocated to an activity, and $c \cdot x_k$, the resulting resource utilization cost. Note that c is just a coefficient. The duration of of an activity is defined as $y_{jk} = w_j/x_k$. Let T_s be the overall project due date, and let c_t be the cost of tardiness per unit of delay. The total cost of tardiness is defined as $c_t \cdot \max\{\, 0, t_m - T_s \,\}$, where t_m is the completion time of the project. Finally, let R_t denote the availability of resource in period t.

We now proceed with the formulation of the integer program. Define two binary variables, β_{jt} and δ_{jt}, corresponding to every $(i, i') \in A$. Also define the following binary variables:

$$\alpha_{jkt} = \begin{cases} 1 & : \text{ if activity } j \text{ is } active \text{ in mode } k \text{ in period } t \\ 0 & : \text{ otherwise} \end{cases}$$

$$z_{jk} = \begin{cases} 1 & : \text{ if activity } j \text{ is executed in mode } k \\ 0 & : \text{ otherwise} \end{cases}$$

Define integer-valued variables as follows:

- s_j: start time of activity j
- f_j: finish time of activity j
- t_i: time of realization of node i

Let T be an upper bound on the project duration. Finally, let the set $\mathcal{T} = \{1, 2, \ldots, T\}$. The complete IP formulation can be stated as follows:

$$\min \sum_j \sum_k z_{jk}(c \cdot x_k \cdot w_j) + c_t \cdot \max\{\,0, t_m - T_s\,\} \tag{3.31}$$

subject to

$$s_j \geq t_i, \qquad \forall j \equiv (i,i') \in A \tag{3.32}$$

$$f_j \geq s_j + \sum_{k=1}^{K} z_{jk} \left(\frac{w_j}{x_k}\right), \ \forall j \tag{3.33}$$

$$t_{i'} \geq f_j, \qquad \forall j \equiv (i,i') \in A \tag{3.34}$$

$$\sum_{k=1}^{K} z_{jk} = 1, \qquad \forall j \in A \tag{3.35}$$

$$\alpha_{jkt} \leq z_{jk}, \qquad \forall j \in A,\ \forall t \in \mathcal{T},\ \forall k \tag{3.36}$$

$$\sum_j \sum_k \alpha_{jkt}\, x_k \leq R_t, \qquad \forall t \in \mathcal{T} \tag{3.37}$$

$$\sum_k \alpha_{jkt} \leq \frac{t - s_j}{T} + 1, \qquad \forall j \in A,\ \forall t \in \mathcal{T} \tag{3.38}$$

$$\sum_k \alpha_{jkt} \leq \frac{f_j - t}{T} + 1, \qquad \forall j \in A,\ \forall t \in \mathcal{T} \tag{3.39}$$

$$\beta_{jt} \geq \frac{t - s_j}{T}, \qquad \forall j \in A,\ \forall t \in \mathcal{T} \tag{3.40}$$

$$\delta_{jt} \geq \frac{f_j - t + 1}{T}, \qquad \forall j \in A,\ \forall t \in \mathcal{T} \tag{3.41}$$

$$\sum_k \alpha_{jkt} \geq \beta_{jt} + \delta_{jt} - 1, \qquad \forall j \in A,\ \forall t \in \mathcal{T} \tag{3.42}$$

$$s, t \geq 0, \text{integer} \tag{3.43}$$

$$\alpha_{jkt},\ \beta_{jt},\ \delta_{jt} \in \{0,1\} \tag{3.44}$$

The objective function (3.31) is composed of total resource cost and tardiness cost[4]. Constraint (3.32) and (3.33) define the start and finish times of activity j, respectively. Constraint (3.34) defines the realization time of node i', $\forall\,(i,i') \in A$. Constraint (3.35) ensures each activity j uses exactly one mode of the resource. Constraint (3.36) maintains consistency between variables α_{jkt} and z_{jk} in terms of resource mode chosen. Finally, Constraint (3.37) enforces resource constraints for every period t.

[4]The term $c_t \cdot \max\{\,0, t_m - T_s\,\}$ is treated as a piecewise-linear function in the model.

Consider the constraints (3.38)–(3.42), which are defined for each combination of time t and activity j. By definition of T, both $\frac{t - s_j}{T}$ and $\frac{f_j - t}{T}$ lie in the interval [-1, 1]. Now consider the following three cases:

1. If $t \leq s_j - 1$, then
   - LHS of (3.38) < 1, forcing α_{jkt} to be 0
   - (3.39) and (3.40) are non-binding
   - (3.41) forces δ_{jt} to be 1
   - (3.42), along with $\alpha_{jkt} = 0$, forces β_{jt} to be 0
2. If $t \geq f_j + 1$, then
   - LHS of (3.39) < 1, forcing α_{jkt} to be 0
   - (3.38) and (3.41) are non-binding
   - (3.40) forces β_{jt} to be 1
   - (3.42), along with $\alpha_{jkt} = 0$, forces δ_{jt} to be 0
3. If $s_j \leq t \leq f_j$, then
   - (3.38) and (3.39) are non-binding
   - (3.40) and (3.41) force both β_{jt} and δ_{jt} to be 1
   - (3.42) now forces α_{jkt} to be 1

3.3 Setup of Computational Experiments for the IP Formulation

In this section we discuss the performance of the IP formulation. We generated the test networks using DAGEN, a software program developed by Agrawal et al. [1996]. DAGEN generates networks according to the desired "complexity index" (CI), which Bein et al. [1992] established as a measure of non-conformity of a graph to the series/parallel topology. For our experiments we generated networks with CI = 2, 3, 4, and 5. For each value of CI, we generated networks with 20, 30, 40, and 50 activities. The integer program was modeled in AMPL and solved using ILOG CPLEX 9.0 on Sun Microsystems Sun Enterprise 220R (UltraSPARC-II 360MHz) with system clock frequency of 120 MHz and 2048 megabytes of RAM. The AMPL code for the IP is included in Appendix B.

We classify the computational experiments into the following categories:

Category 1: Fix the work content of all activities, the value of resource availability (R), and the available modes to choose from. For each value of n — that is, the number of activities — repeat the experiment for

(a) different values of complexity index (CI) of the network, and
(b) different values of the upper bound on the project completion time(T).

Category 2: Fix the upper bound on the project completion time (T) and the number of activities (n). Repeat the experiment for

(a) different values of resource availability (R), and
(b) different values of complexity index (CI) of the network.

The results in Category 1 shed light on how the performance of the optimization of the IP is affected with the increase in the size and complexity of the project network. The results in Category 2 help us in learning how the performance of the IP optimization is affected with the restrictiveness of the resource availability. The computational results are reported in the following sections.

3.3.1 Computational Results: Category 1

We report the computation times for the following problem parameters:

(i) R = 5, resource modes = { 1, 2, 3 }, n = 20, 30
(ii) R = 7, resource modes = { 1, 2, 3 }, n = 40, 50

The resource availability was increased from 5 to 7 for n = 40 and n = 50 for the following reason: the restrictiveness of the resource availability coupled with the increase in number of activities renders the IP infeasible. We have to either reduce the work content of the activities or increase the upper bound T.

It is clear from Figs. 3.4–3.7 that as the upper bound on project completion T increases, the computation times increase significantly. A steeper increase in computation times is evident when the CI changes from 3 to 4 than when the CI changes from 2 to 3.

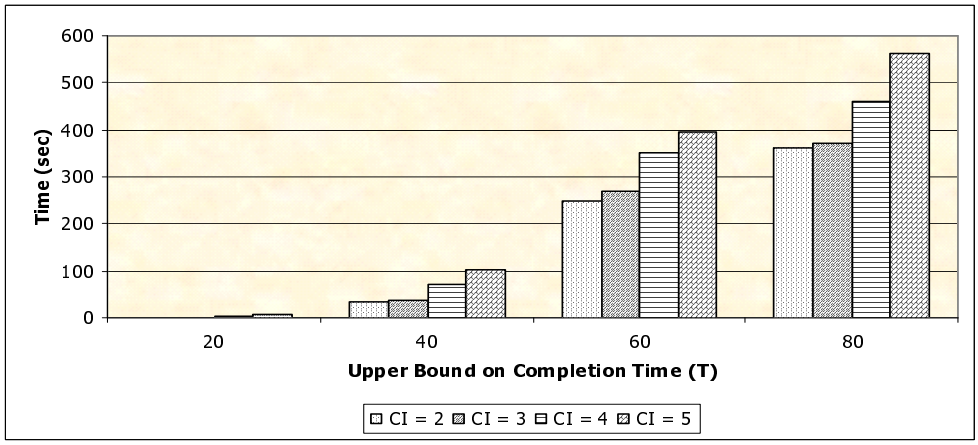

Figure 3.4: Comparison of Computation Times: R = 5, n = 20, T = 20, 40, 60, 80.

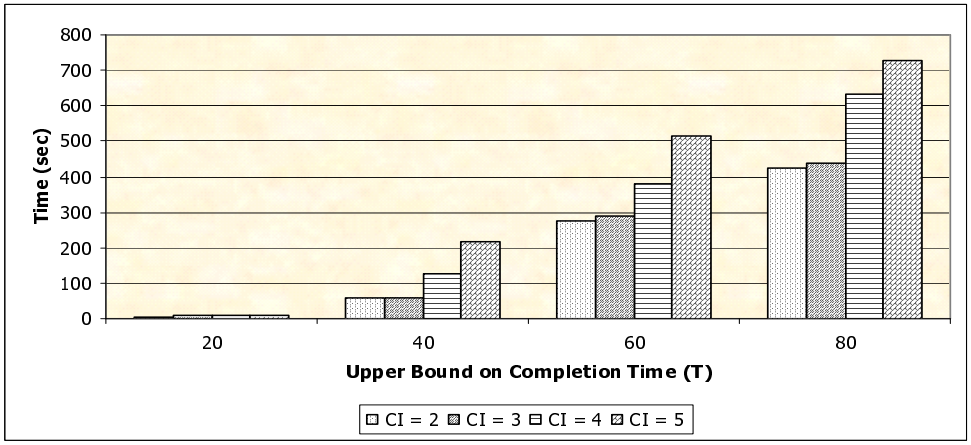

Figure 3.5: Comparison of Computation Times: R = 5, n = 30, T = 20, 40, 60, 80.

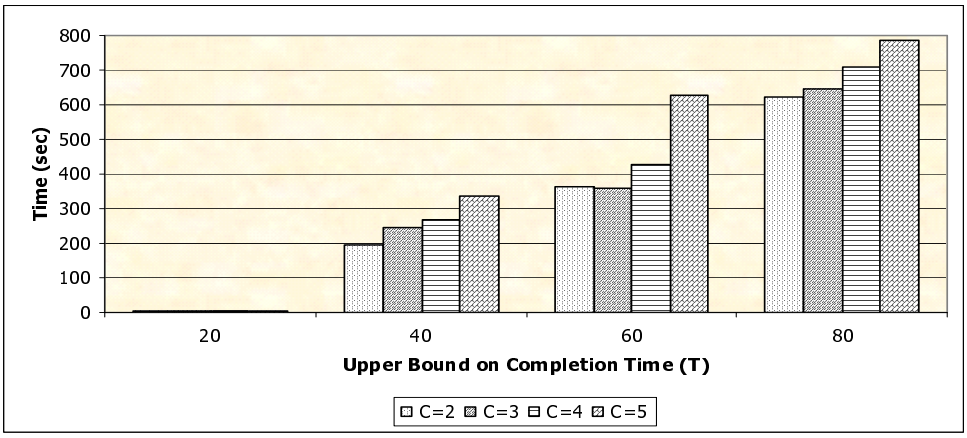

Figure 3.6: Comparison of Computation Times: R = 7, n = 40, T = 20, 40, 60, 80.

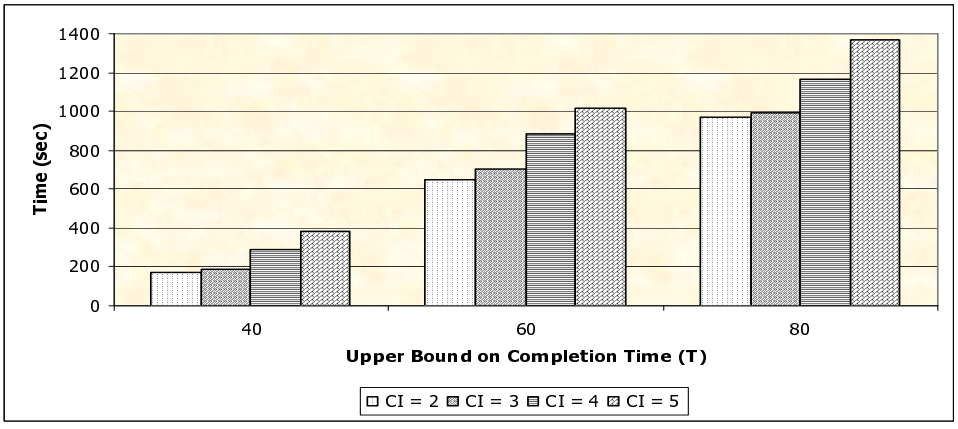

Figure 3.7: Comparison of Computation Times: R = 7, n = 50, T = 40, 60, 80.

3.3.2 Computational Results: Category 2

In order to observe the performance of the IP with relaxing the resource availability, we chose $n = 40$, $T = 40$ and varied the resource level R from 7 to 10 in increments of 1. As seen from Fig. 3.8, the computation time drop drastically as R increases. The decrease is steeper in networks with higher CI (4 and 5) mainly because of more parallel paths in the network.

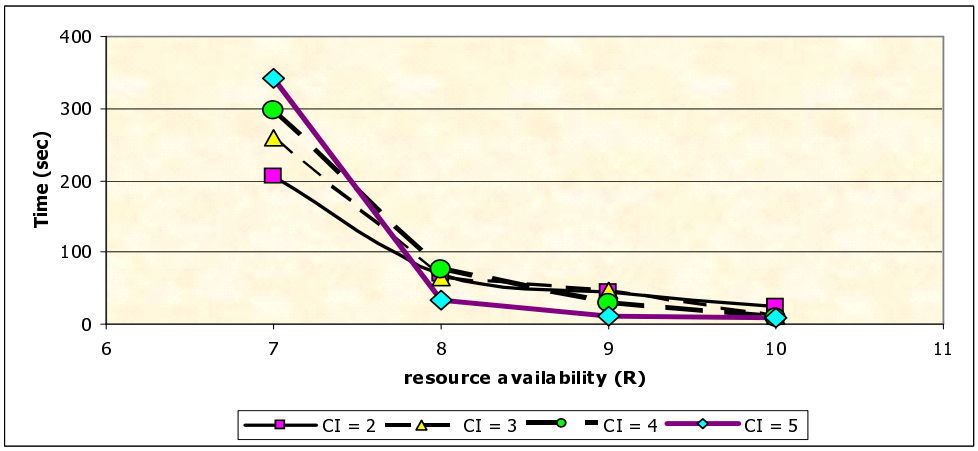

Figure 3.8: Comparison of Computation Times: $T = 40$, n = 40, $R = 7, 8, 9, 10$.

3.3.3 Observations from Computational Experiments

From the Category 2 computational experiments, it is clear that resource availability plays an important role in the computation effort needed to solve the IP. This is because of the following reason: when there is scarcity of resources, it limits the number of activities that can be processed in parallel. This means that, some activities that could have been processed in parallel now will be processed sequentially, thereby prolonging the project. Since the cardinality of the set $\mathcal{T}$ depends on the upper bound on the project completion time T, it is natural that with increase in T, the problem size increases accordingly.

Chapter 4

Optimal Resource Allocation – Stochastic Case

In this chapter we study the optimal resource allocation to the activities of a project in order to optimize an economic objective in the face of uncertainty. In Section 4.1 we state our problem along with the assumptions. In Section 4.2 we outline a Policy Iteration–like procedure for solving such networks. We model it as a Continuous-Time Markov Chain on the lines of the work by Kulkarni and Adlakha [1986] and we proceed to introduce the phase type distribution that plays an important role in the policy evaluation. In Section 4.3 we outline the technique of stochastic programming (SP), which is also known as mathematical programming under uncertainty. We discuss a recent research effort that applies the concept of SP to a stochastic resource-constrained project scheduling problem. Following that discussion, we demonstrate the inapplicability of standard SP techniques to our problem. Finally in Section 4.4 we describe a simulation-cum optimization technique that implements variance reduction techniques (VRT) in order to reduce the sample size and the sample variance. We illustrate the effectiveness of the VRT implemented with the help of an example, and also briefly describe how the output data of the simulation experiment can be used to address related problems of interest.

4.1 Problem Statement and Assumptions

We are given a project network of n activities defined by a graph $G = (N, A)$, in which A is the set of arcs defining the activities and the precedence relations among them (AoA mode of representation). Let W_j denote the *work content* of activity j. We consider a single

renewable resource and let x_j be the allocation of the resource to activity j. Assume R as the total availability of the renewable resource in any period. Define c_r to be the per-unit cost of usage of resource per unit time. We further assume that the project has a specified due date T_s, a constant, and let $c_t\,(\max\{\,0, \Upsilon - T_s\,\})$ be the cost of tardiness when the project completes at time Υ, a random variable. Our objective is to determine the resource allocation vector, $\mathbf{X}$ to the activities such that the overall *expected* cost of resource allocation and tardiness is minimized.

4.2 Policy Iteration–like Approach

For the sake of tractability in analysis, assume that the activity's work content, W_j, a random variable, follows an exponential distribution

$$W_j \sim \text{Exp}(\lambda_j),\ \forall j \in A.$$

Assume that the resource allocation x_j is bounded from above and below as:

$$0 \leq l_j \leq x_j \leq u_j < \infty,\ \forall j \in A.$$

Let the resulting duration of activity j,

$$Y_j = \frac{W_j}{x_j},$$

It is clear that Y_j is also exponentially distributed, but with parameter $x_j\lambda_j$, henceforth denoted by μ_j.

Assume the total cost of resource allocation to activity j is quadratic in the allocation over the duration of the activity; i.e.,

$$C_j = c_r\, x_j^2\, Y_j = c_r\, x_j\, W_j.$$

In this section we analyze the problem from the perspective of "managerial flexibility." This represents the main departure from the work of Tereso et al. [2001, 2003a,b]. Assume further that it is possible to change the resource allocation to the same activity according to the changes in the state of the project. To be more precise, consider the scenario where more than one activity are in progress. Eventually one of them finishes first. Now the manager

is allowed to change the allocation of the resource(s) to the still ongoing activities. This perspective requires the definition of the "state" of the project and the corresponding "state space" of the project network. This is basically the model treated by Kulkarni and Adlakha [1986], which is based on the realization that we are in fact dealing with a Continuous-Time Markov Chain (CTMC). The main distinction between this analysis and that reported in their work is that we are interested in *optimization* of resource allocation, while they dealt only with the *analysis* of the resulting CTMC.

4.2.1 Continuous-Time Markov Chain

In order to be able to transform our problem into a CTMC, we introduce some notation from Kulkarni and Adlakha [1986].

We now have $G = (N, A)$ to be a PERT network. We assume it starts at time zero and ends at time Υ, a random variable. During the course of project execution, each activity can be in one and only one of the following three states:

(i) *active*: an activity a is *active* at time t if it is being executed at time t.

(ii) *dormant*: an activity a is *dormant* at time t if it has finished but there is at least one unfinished activity that ends at the same node as a.

(iii) *idle*: an activity a is called *idle* at time t if it is neither active nor dormant at time t. In other words, the activity is either completed or is yet to be started.

For $t \geq 0$, define the sets

$$\begin{aligned} \mathcal{A}(t) &= \{j \in A : j \text{ is active at time } t\}, \\ \mathcal{D}(t) &= \{j \in A : j \text{ is dormant at time } t\}, \\ \mathcal{P}(t) &= (\mathcal{A}(t), \mathcal{D}(t)) : \text{ the state at time } t. \end{aligned}$$

We also have the following assumption:

A1. All activity work content, and hence the durations, are mutually independent positive exponential random variables.

Let $\mathcal{S}$ denote the set of all $\mathcal{P}(t)$, and let $\overline{\mathcal{S}} = \mathcal{S} \cup \{(\emptyset, \emptyset)\}$. Note that $\mathcal{P}(t) = (\emptyset, \emptyset)$ implies that all activities are idle at time t and hence the project is completed[1].

Under the assumption $A1$, $\{\mathcal{P}(t), t \geq 0\}$ is a CTMC on $\overline{\mathcal{S}}$. Furthermore we make the following observations:

1. The state $(\emptyset, \emptyset)$ is absorbing. This means that, once the project is completed, it remains completed.

2. $\{\mathcal{P}(t), t \geq 0\}$ visits any state in $\mathcal{S}$ at most once, i.e. each activity is executed exactly once. This implies that all states in $\mathcal{S}$ are transient.

Finally, since we have a project with finite number of activities, and correspondingly a finite state space, $\{\mathcal{P}(t), t \geq 0\}$ is a finite-state, absorbing CTMC with a single absorbing state $(\emptyset, \emptyset)$.

We are now ready to introduce the *phase type* distribution and its properties.

4.2.2 Phase Type Distribution

The name "phase type" distribution stems from the fact that an Erlang distribution is derived as the sum of "stages" or "phases", all exponentially distributed with the same parameter λ. The generalized Erlang distribution of order *r* has *r* phases (stages) each is exponentially distributed but with possibly different parameters $\lambda_1, \cdots, \lambda_r$.

Definition 4.1. *A continuous probability distribution $F(\cdot)$ is of the phase type (PH-distribution) if it is the distribution of the time until absorption in a finite-state Markov process with a single absorbing state; that is, there exists a probability vector $(\boldsymbol{\alpha}, \alpha_{r+1})$ and an infinitesimal generator matrix of the form*

$$\mathbf{Q} = \begin{bmatrix} \mathbf{T}_{r\times r} & \mathbf{T}^0_{r\times 1} \\ \mathbf{0}_{1\times r} & 0_{1\times 1} \end{bmatrix} = \begin{bmatrix} \mathbf{T} & -\mathbf{Te} \\ \mathbf{0} & 0 \end{bmatrix} \tag{4.1}$$

where $\mathbf{e}$ is a $r \times 1$ vector of ones, $T_{i,i} < 0$ for $1 \leq i \leq r$, and $T_{i,j} \geq 0$ for $i \neq j$.

[1] Note that at $t = 0$, i.e., when the project is not yet started, we have $\mathcal{P}(t) = (\emptyset, \emptyset)$.

Also,

$$\mathbf{Te} + \mathbf{T}^0 = 0,$$

and the initial probability vector (the so-called 'counting probability') of $\mathbf{Q}$ is given by the vector $(\boldsymbol{\alpha}, \alpha_{r+1})$, with

$$\boldsymbol{\alpha} \cdot \mathbf{e} + \alpha_{r+1} = 1.$$

The pair $(\boldsymbol{\alpha}, \mathbf{T})$ is called '*representation*' of $F(\cdot)$. In our case, the process starts in state 1 with probability 1. States $1, \ldots, r$ are transient so that absorption into state $r+1$ from any initial state is certain.

The first result quoted by Neuts [1989] is that the matrix $\mathbf{T}$ is nonsingular (a necessary and sufficient condition for the states $1, \ldots, r$ to be transient). Observe that

$$\mathbf{T}^k \longrightarrow \mathbf{0} \text{ as } k \longrightarrow \infty.$$

Assuming an initial probability vector $(\boldsymbol{\alpha}, \alpha_{r+1})$, the c.d.f. of the time to absorption in state $r+1$ corresponding to the initial probability vector $(\boldsymbol{\alpha}, \alpha_{r+1})$ is given by

$$F(z) = \mathbf{1} - \boldsymbol{\alpha} \cdot e^{\mathbf{T}z} \cdot \mathbf{e}, \text{ for } z \geq 0 \tag{4.2}$$

We make the following observations about the properties of the distribution $F(\cdot)$:

(a) It has a jump of height α_{r+1} at $z = 0$. Evidently this is the probability that the process starts in the absorbing state. This case is of no concern to us since it implies that the project is complete at its start, which would imply that there is no project.

(b) Its density portion $f(z) = F'(z)$ on $(0, \infty)$ — i.e., excluding the 0 point — is given by

$$f(z) = F'(z) = \boldsymbol{\alpha} \cdot e^{\mathbf{T}z} \cdot \mathbf{T}^0 \tag{4.3}$$

In our case the "portion" of the domain of *x* is the whole real line including the point at the origin because $\alpha_{r+1} = 0$.

(c) The Laplace-Stieltjes transform $F(s)$ of $F(\cdot)$ is given by

$$F(s) = \alpha_{r+1} + \boldsymbol{\alpha}(s\mathbf{I} - \mathbf{T})^{-1}\mathbf{T}^0, \text{ for } \mathrm{Re}(s) \geq 0 \tag{4.4}$$

The first term is the multiplier of $e^{-sz}|_{z=0}$, which in our case does not exist.

(d) The noncentral moments (about the origin) ϕ_i' of $F(\cdot)$ are all finite and given by

$$\phi_i' = (-1)^i \times i! \left(\boldsymbol{\alpha}\mathbf{T}^{-i}\mathbf{e}\right), \text{ for } i \geq 0 \tag{4.5}$$

4.2.3 Details of the Policy Iteration–like Procedure

The procedure we outline is an *approximation* in the sense that we are considering the *cost of tardiness of the expected* project completion time, which we know is *not* equal to the *expected cost of tardiness* of the project completion time.

We wish to discuss this procedure for the following reason — we know that the exponential distribution is an *over-estimate* of all NBUE[2]-distributions, which constitute the class of distributions we encounter in the field of project networks. We also know (from Jensen's inequality) that the cost of the expected tardiness is an *under-estimate* of the expected cost of tardiness. In other words, let $g(\cdot)$ be a convex function representing tardiness cost. We know that the tardiness value depends on the project completion time, which is a random variable, Υ. For the sake of convenience, let $g(\cdot) = c_t \max\{0, \Upsilon - T_s\}$. Then, from Jensen's inequality, we have the following:

$$c_t \cdot \max(0, \mathbb{E}[\Upsilon - T_s]) \leq \mathbb{E}[g(\cdot)].$$

4.2.3.1 Illustrative example

Consider a simple activity network comprising three activities as shown in Fig. 4.1. The actual network is at the top left corner and the corresponding state transition diagram is displayed at the bottom.

Assume the unit cost of resource usage (c_r) is normalized at 1.0 and the cost of tardiness (c_t) is 3.0. Let the work content of the activities 1, 2, and 3 be exponentially distributed with parameters 0.2, 0.1, and 0.07, respectively. Let the lower and upper bounds on resource availabilities be 1 and 4, respectively, for all activities;i.e., $1 \leq x_j \leq 4$, for $j = 1, 2, 3$. Finally let the due date T_s be 8.

Initialization:

Assume we start at the initial resource allocation vector $\mathbf{x^0} = [1, 1, 1]$. The corresponding

[2]NBUE refers to New Better than Used in Expectation

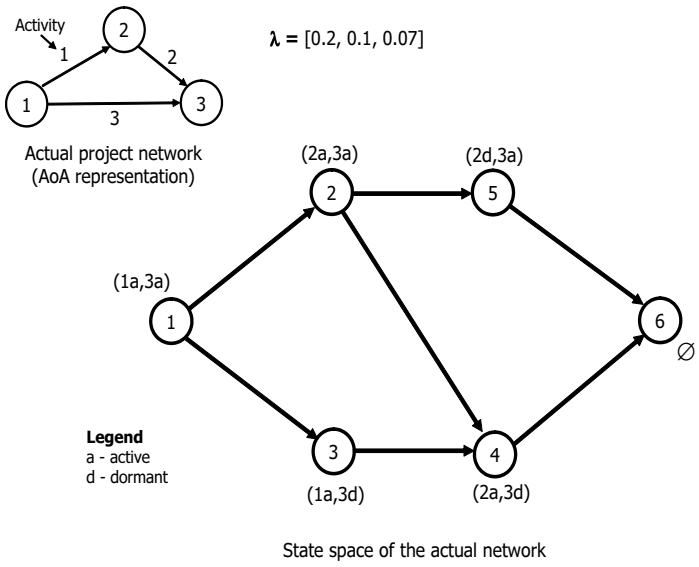

Figure 4.1: Example Network 2 and its State Space

infinitesimal generator matrix,

$$\mathbf{Q_0} = \begin{bmatrix} -0.27 & 0.2 & 0.07 & 0 & 0 & 0 \\ 0 & -0.17 & 0 & 0.07 & 0.10 & 0 \\ 0 & 0 & -0.2 & 0.2 & 0 & 0 \\ 0 & 0 & 0 & -0.10 & 0 & 0.10 \\ 0 & 0 & 0 & 0 & -0.07 & 0.07 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The expected project completion time and various costs are computed as follows:

$$\begin{aligned} \mathbb{E}(\Upsilon_3) &= 21.22 \ \text{(obtained using Eqn. 4.5)} \\ \text{resource cost} &= 1 \times \left(\frac{1}{0.2} + \frac{1}{0.1} + \frac{1}{0.07} \right) = \$29.29 \\ \text{tardiness cost} &= 3 \times (21.22 - 8) = \$39.67 \\ \text{total cost} &= \$68.96 \end{aligned}$$

Achieving Cost Reduction:

We can seek improvement in the total cost by one of the following two methods.

(a) Increase the allocation x_1 in small increments Δ (say $\Delta = 0.1$) until no further improvement in the objective value is achieved. We know that an increment Δ in the resource allocation to activity 1 changes the rate matrix $\mathbf{Q_0}$ to

$$\mathbf{Q_1} = \begin{bmatrix} -0.27-\Delta & 0.20+\Delta & 0.07 & 0 & 0 & 0 \\ 0 & -0.17 & 0 & 0.07 & 0.10 & 0 \\ 0 & 0 & -0.20-\Delta & 0.20+\Delta & 0 & 0 \\ 0 & 0 & 0 & -0.10 & 0 & 0.10 \\ 0 & 0 & 0 & 0 & -0.07 & 0.07 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

from which we can obtain the expected project duration using Eqn. (4.5) and compute the other costs.

Using this functional evaluation one can adopt a suitable iterative scheme incorporating a *steepest-descent* search over possible values of resource allocations for all activities. In other words,

$$\text{“Best” policy, } \mathbf{x}_{\text{best}} = \underset{\mathbf{x}}{\text{argmin}}\ c_r \left(\sum_{j \in A} x_j \mathbb{E}\left[W_j\right] \right) + c_t \left(\max\{\, 0, E[\Upsilon] - T_s \,\} \right)$$

(b) Because of the symmetry between the two paths in the (original) project network ($\pi_1 = 1, 2$ and $\pi_2 = 3$), suppose we augment the resource allocation to activity 1 so that the expected lengths of the two paths are equal; i.e., we seek the solution to the equation

$$\begin{aligned} \frac{1}{0.2x_1} + \frac{1}{0.1} &= \frac{1}{0.07} \\ \text{i.e., } \frac{5}{x_1} &= \frac{1}{0.07} - 10 = 4.2857 \\ \text{which} \quad \Rightarrow \quad x_1 &= \frac{5}{4.2857} = 1.1667 \end{aligned} \tag{4.6}$$

Now, select a value the x_1 large enough to make the cost larger than at $x_1 = 1.1667$, then use dichotomous search to find the optimal value. Say for instance, we select

$x_1 = 1.5$. The new allocation vector $\mathbf{x^1} = [1.5, 1, 1]$ results in the following infinitesimal generator matrix:

$$\mathbf{Q_1'} = \begin{bmatrix} -0.37 & 0.3 & 0.07 & 0 & 0 & 0 \\ 0 & -0.17 & 0 & 0.07 & 0.10 & 0 \\ 0 & 0 & -0.3 & 0.3 & 0 & 0 \\ 0 & 0 & 0 & -0.10 & 0 & 0.10 \\ 0 & 0 & 0 & 0 & -0.07 & 0.07 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The corresponding expected project completion time and the relevant costs are computed as follows.

$$\begin{aligned} \mathbb{E}(\Upsilon) &= 20.15 \text{ (obtained using Eqn. 4.5)} \\ \text{resource cost} &= 1 \times \left(1.5 \times \frac{1}{0.2} + \frac{1}{0.1} + \frac{1}{0.07}\right) = \$31.79 \\ \text{tardiness cost} &= 3 \times (20.15 - 8) = \$36.44 \\ \text{total cost} &= \$68.23 \end{aligned}$$

Notice that the total cost has decreased. Let us now continue to increase the allocation to activity 1 further, say to 1.75. The updated completion time and the relevant costs corresponding to the allocation $\mathbf{x^2} = [1.75, 1, 1]$ are:

$$\begin{aligned} \mathbb{E}(\Upsilon) &= 19.86 \text{ (obtained using Eqn. 4.5)} \\ \text{resource cost} &= 1 \times \left(1.75 \times \frac{1}{0.2} + \frac{1}{0.1} + \frac{1}{0.07}\right) = \$33.04 \\ \text{tardiness cost} &= 3 \times (19.86 - 8) = \$35.58 \\ \text{total cost} &= \$68.62 \end{aligned}$$

Notice that the total cost is larger than the total cost obtained using $\mathbf{x^1}$. Although the tardiness cost has been reduced, the reduction has been more than offset by the increase in the resource cost. Therefore we know that the maximal gain in cost is for some x_1 in the interval $[1.50, 1.75]$. Depending on the desired accuracy, one may either enumerate the values of x_1 in this interval, or use dichotomous search, or even use Fibonacci search to determine the optimum.

Having secured the most improvement from activity 1, we now try changing the allocation to activity 2 (or 3). If successful, we follow the same procedure with activity 3 (or 2), and so on until no further improvement is possible. Note that subsequent reduction in cost due to changes in the allocation to either activities 2 or 3 may induce further improvement in cost in activity 1.

Following through with the iterations, the "best policy" for the example network is $\mathbf{x}^* = [1.5, 1.5, 1.5]$ with an associated cost of 62.38. Coincidentally, this is also seen in the Table 4.1 below, which actually summarizes a part of the solution search process. The table displays values of total cost as the allocation for activity 3, x_3, is varied while keeping the allocations for activities 1 and 2, (x_1, x_2), fixed. The surface plot in Fig. 4.2 summarizes the table graphically.

Table 4.1: Values of Expected Total Cost for Fixed Allocations (x_1, x_2) and Varying x_3

		x_3								
		1	1.25	1.5	1.75	2	2.25	2.5	2.75	3
(x_1, x_2)	(1, 1)	68.96	66.58	66.57	67.83	69.82	72.28	75.04	78	81.11
	(1.25,1.25)	66.5	63.55	63.11	64.05	65.8	68.06	70.67	73.5	76.5
	(1.5,1.5)	66.48	63.12	62.38	63.07	64.63	66.74	69.21	71.94	74.86
	(1.75,1.75)	67.76	64.11	63.13	63.64	65.04	67.02	69.39	72.04	74.88
	(2,2)	69.8	65.93	64.77	65.13	66.41	68.29	70.57	73.15	75.93
	(2.25,2.25)	72.32	68.27	66.97	67.21	68.4	70.19	72.41	74.92	77.65
	(2.5,2.5)	75.16	70.97	69.56	69.7	70.8	72.53	74.68	77.14	80.36
	(2.75,2.75)	78.2	73.91	72.4	72.47	73.5	75.17	77.28	80.54	84.11
	(3,3)	81.4	77.02	75.44	75.44	76.42	78.04	80.71	84.29	87.86

In proposing any sequential search procedure one must establish two important properties:

(a) the process terminates finitely (for any desired accuracy) at the desired optimum.

(b) at termination, the *policy* obtained and its corresponding *value* are *independent of the sequence* in which the activities were selected. (For instance in the above example, had we selected activity 2 at the start then proceeded from there, would we still have

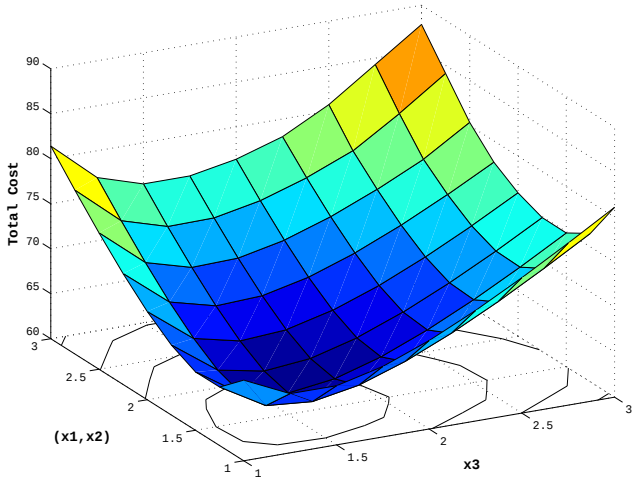

Figure 4.2: Surface Plot of Resource Allocations versus Total Cost

reached the same conclusion?)

These two propositions would be easily established if one can demonstrate that the reward function

$$\begin{aligned} z\left(\{x_j\}_{j=1}^{|A|}\right) &= \mathbb{E}\left[\sum_{j\in A} c_r x_j W_j + c_t \cdot \max\{0, \Upsilon - T_s\}\right] \\ &= c_r \sum_{j\in A} x_j \mathbb{E}\left[W_j\right] + c_t \cdot \mathbb{E}\left[\max\{0, \Upsilon - T_s\}\right] \end{aligned} \tag{4.7}$$

is convex in the ensemble of decision variables $\{x_j\}$. The first term in (4.7) is linear in x_j, and therefore convex. It is the second term that is not straightforward because the impact of changes in x_j, $l_j \leq x_j \leq u_j$, on the project completion time Υ is intertwined with other allocations through the inverse of the matrix $\mathbf{T}$ (in Eqn. 4.1).

However, one thing is definitely clear: the function $\max\{0, \Upsilon - T_s\}$ is monotone non-increasing in each x_j. In other words, under strict precedence relations as dealt with in our thesis, an increase in x_j from its lower bound can lead only to a shift in the distribution of Υ towards 0, or stay unchanged. Therefore the function $c_t \cdot \mathbb{E}\left[\max\{0, \Upsilon - T_s\}\right]$ of Eqn. 4.7 is monotonically non-increasing in its argument up to a point (or ridge, or surface), beyond which it is monotonically non-decreasing due to increase in resource cost with tardiness staying constant at 0 (see Fig. 4.2). If the increment Δ_j is chosen "small enough" then the

unique minimal *value* of $z(\cdot)$ must be achieved. Observe that there may be a large number of alternate optimal *policies* $\{x_j\}$ that yield the same *value.* Finally the fact that the decision variables are bounded on both sides ensures the finiteness of the procedure.

The above argument establishes the validity of the following proposition.

Proposition 4.1. *For the choice of the increment Δ_j sufficiently small, the iterative scheme of Policy Iteration achieves the optimal value and an optimal policy in a finite number of steps.*

4.2.4 Limitations of the PI-like Approach

We summarize some of the limitations of the "Policy Iteration" procedure described previously.

- In the worst case scenario, if G is a completely directed acyclic network (DAG) with n nodes and $n(n-1)/2$ arcs, then the state space for G is given by $N(n) = J_n - J_{n-1}$, where
$$J_n = \sum_{k=0}^{n} 2^{k(n-k)}$$
- The solution search space increases exponentially. Assume we have 10 activities, and if we intend to discretize the resource availability interval into 5 points. We are now potentially looking at 5^{10} points!
- The functional evaluation applies only to the case when all activity work contents are independent exponential random variables, in which case we are able to
  - secure a CTMC, and
  - change resource allocation in mid-stream for any activity without any change in distribution of the "remaining" work content.
- We cannot implicitly incorporate the aggregate resource constraints; they have to be done explicitly as part of a search procedure.

4.3 Overview of Stochastic Programming

Stochastic programming is a framework for modeling optimization problems that involve uncertainty. The most widely applied and studied stochastic programming models are two-stage linear programs. Here the decision maker takes some action in the first stage, after which a random event occurs affecting the outcome of the first stage. A recourse decision can then be made in the second stage that compensates for any bad effects that might have been experienced as a result of the first-stage decision. The optimal policy from such a model is a single first-stage policy and a collection of recourse decisions (a decision rule) defining which second-stage action should be taken in response to each random outcome. Mathematically, this can be stated as:

$$\left\{ \begin{array}{llllll} \min & c^{\mathrm{T}}x & + & \mathbb{E}_{\omega}[Q(x,\omega)] \\ \text{s.t.} & Ax & = & b \\ & l & \leq & x & \leq & u \end{array} \right\} \tag{4.8}$$

where

$$\left\{ \begin{array}{llllll} Q(x,\omega) & = & \min & d(\omega)^{\mathrm{T}}y \\ & & \text{s.t.} & T(\omega)x & + & W(\omega)y & = & h \\ & & & & & y & \geq & 0 \end{array} \right\} \tag{4.9}$$

The first linear program of (4.8) minimizes the first-stage direct costs, $c^{\mathrm{T}}x$ plus the expected recourse cost $\mathbb{E}_{\omega}[Q(x,\omega)]$, over all of the possible scenarios of the second stage while meeting the first-stage constraints, $Ax = b$. The recourse cost Q depends both on x, the first-stage decision, and on the random event ω. The second LP of (4.9) describes how to choose $y(\omega)$ (a different decision for each random scenario). It minimizes the cost $d^{\mathrm{T}}y$ subject to some recourse function, $Tx + Wy = h$. One important thing to notice in stochastic programs is that the first-stage decision, x, is independent of which second-stage scenario actually occurs. This is called the nonanticipatory property.

Solution approaches to stochastic programming models are driven by the type of probability distributions governing the random parameters. A common approach to handling uncertainty is to define a small number of scenarios to represent the future. In this case it is possible to compute a solution to the stochastic programming problem by solving a

deterministic equivalent linear program represented by (4.10) by introducing a different second-period y variable for each scenario.

$$\left\{\begin{array}{lrcl} \min & c^{\mathrm{T}}x + \displaystyle\sum_{i=1}^{L} p_i d_i^{\mathrm{T}} y_i & & \\ \text{s.t.} & Ax = & b & \\ & T_i x + W_i y_i = & h_i, & \forall\, i = 1, \ldots, L \\ & x \geq & 0 & \\ & y_i \geq & 0 & \forall\, i = 1, \ldots, L \end{array}\right\} \tag{4.10}$$

where p_i is the probability of occurrence of scenario i and L represents the total number of scenarios chosen. Notice that the nonanticipatory constraint is met in $Ax = b$. There is only one first-stage decision, x, whereas there are L second-stage decisions, one for each scenario. The first-period decision cannot prefer one scenario over another and must be feasible for each scenario — that is, $Ax = b$ and $T_i x + W_i y_i = h_i$ for $i = 1, \ldots, L$. Since we solve for all the decisions, x and y_i simultaneously, we are choosing x to be (in some sense) optimal over all the scenarios.

These problems are typically very large scale LP problems, and so, much research effort in the stochastic programming community has been devoted to developing algorithms that exploit the problem structure, in particular in the hope of decomposing large problems into smaller more tractable components. Here convexity is a key property.

When the probability distributions of the random parameters are continuous, or there are many random parameters, one is faced with the problem of constructing appropriate scenarios to approximate the uncertainty. One approach to this problem constructs two different deterministic equivalent problems, the optimal solutions of which provide upper and lower bounds on the optimal value of the original problem.

An alternative solution methodology replaces each of the random variables by a finite random sample and solves the resulting (deterministic) mathematical programming problem as one would do for the finite scenario case. This is often called an *external sampling method*. Under fairly mild conditions one can obtain a statistical estimate of the optimal solution value that converges to the optimal solution as the sample size increases.

Stochastic integer programming models arise when the decision variables are required to take on integer values. In most practical situations this entails a loss of convexity

and makes the application of decomposition methods problematic. Techniques for solving stochastic integer programming models are an active research area.

4.3.1 Application of Stochastic Programming in Stochastic Project Scheduling

Stochastic programming techniques have been applied to projects with probabilistic activity durations. However, the literature is very sparse for project scheduling under resource constraints, in which case the focus is on minimizing an economic objective. A notable mention is the work by Valls et al. [1998]. We briefly discuss their model because it provides a good understanding of the requirements for stochastic programming to be valid.

They deal with the problem of resource-constrained project scheduling with stochastic activity interruptions, where they minimize the total expected value of weighted tardiness. Further details of the problem are summarized as follows:

- The project, represented by a directed graph $G = (A, E)$, where A is the set of activities and E is the set of precedence relationships.
- The set of activities, $A = DA \cup SA$, where
  - DA: set of deterministic activities
  - SA: set of stochastic activities, i.e., those that are interrupted for an uncertain amount of time (initial processing time is known; length of the interruption and the final processing time are uncertain).
- There are $|R|$ renewable resource types. The availability of each resource type k in each time period is R_k units, for $k = 1, \ldots, |R|$.
- Each activity i requires r_{ij} units of resource j for its completion.
- The two parts of the stochastic activities in SA, namely before and after the interruption, require the same number of units of each resource.
- The processing time of the second part (after the interruption) of a stochastic activity is independent of the length of the interruption.

From the scheduling perspective, three different situations define a new decision stage, namely

(a) beginning of the project,

(b) completion of an activity $\in DA$ or completion of first part of activity $\in SA$, and

(c) end of an interruption.

At each of the three above stages, one must consider the available resources, the activities yet to be scheduled, and the precedence relationships in order to make scheduling decisions. Valls et al. [1998] transform this problem into a two-stage decision problem, where, in the first stage they assign a priority to each activity and in the second stage, upon resolution of the uncertainties of the interruptions, they construct a schedule using these priorities. Therefore a valid solution representation is any topological ordering of G.

Their objective can be stated mathematically as

$$\min \ ET(\pi) = \sum_{s \in L} p_s T_w(\pi)$$

where p_s is the probability associated with scenario s and T_w represents the total weighted tardiness for scenario s – that is,

$$T_w(\pi) = \sum_{j \in A} \lambda_i \max\{c_{js} - d_j, 0\},$$

where d_j is the due date of activity j and c_{js} is the completion time of activity j under scenario s. Since the completion time of each activity depends on the ordering π, both T_w and expected tardiness ET also depend on π.

The solution procedure implemented by Valls et al. [1998] is a hybrid algorithm based on scatter search techniques. The procedure searches for improved solutions by generating priority orderings and evaluating the quality of the ordering by scheduling the activities under each scenario.

4.3.2 Inapplicability of Stochastic Programming to Our Problem

As seen from the previous section, the procedure of Valls et al. [1998] essentially mimics a two-stage decision making process. In the first stage the priorities are set and in the second stage there is a recourse available for adjusting the schedule according to the resolution of the uncertainties. The nonanticipatory property is retained at all times because the priorities set in the first stage are not disturbed.

We now present a brief discussion to illustrate why a stochastic programming approach, similar to the one just presented, cannot be applied to our problem. Consider a simple network (AoA representation) as shown in Fig. 4.3.

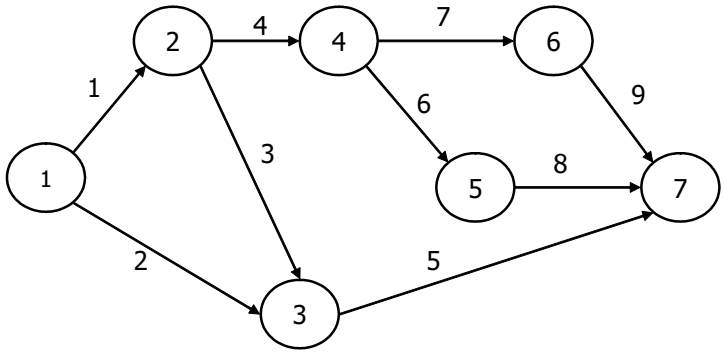

Figure 4.3: Example Network 3.

We can enumerate the *udc*'s and depict them in a hierarchial form as shown in Fig. 4.4. The numbers within the node represent the activities in a *udc* and the number on the arc indicates the completed activity. An explanation on how to interpret the "tree" in Fig. 4.4 follows: consider the nodes (in the "tree") {1, 2} and {2, 3, 4} joined by an arrow with the number 1 on it. This means that when activity 1 in *udc* {1, 2} is completed, the "active" *udc* now becomes {2, 3, 4}. On the other hand if activity 2 finishes first, then activities in {3, 4} have to wait until the preceding activity, namely activity 1, is completed.

The first problem encountered is the identification of a "stage." A stage can be thought of as an epoch in time when one event has completed and the next event can start. In our case, a logical possibility would be to consider activities processed in a *udc* as a stage. In other words, "processing" one *udc* after another would translate into proceeding stage-wise. As one can see, this itself presents a very daunting challenge — currently, most stochastic programs are modeled as two-stage programs. Modeling and solving multi-stage stochastic programs is very hard and is an active area of research.

Besides the modeling difficulty, there is another issue that needs to be considered — the nonanticipatory property. In other words, the first-stage decision; namely the assignment of the priority π, is independent of which scenarios occur in the subsequent stages. Defining and interpreting what a decision is in a given period corresponds to is unclear, thereby making it even more difficult to try and ensure the nonanticipatory property. In order to understand the intricacy of the problem, consider a particular scenario related to the example project in Fig. 4.3. Assume there is an algorithm designed for adaptive resource

allocation — that is, it proceeds stage-by-stage, while performing some sort of "look ahead" action in allocation of resources. The tree shown in Fig. (4.4) reveals an interesting problem that arises in this scheme of adaptive resource allocation. Assume that, in the first "stage", we have obtained (x_1^*, x_2^*) that minimizes the *expected* total cost. We now "fix" the allocation of these 2 activities and proceed to obtain the optimal values for (x_3, x_4). Now, when we sample the work content for activities {1, 2, 3, 4}, we know, based on (x_1^*, x_2^*) obtained in the previous step, whether node 2 is realized first or node 3. If node 2 is realized first we have to optimize (x_3, x_4) over the updated resource availability, namely $R - x_2$. On the other hand, if node 3 is realized first we have to wait until activity 1 finishes, then optimize (x_3, x_4) over the original resource availability R.

Here comes the dilemma — we have optimized (x_3, x_4) over 2 different values of resource availabilities. How do we reconcile them? This will be the case whenever we have two arcs terminating at any given node in the cutset hierarchy tree.

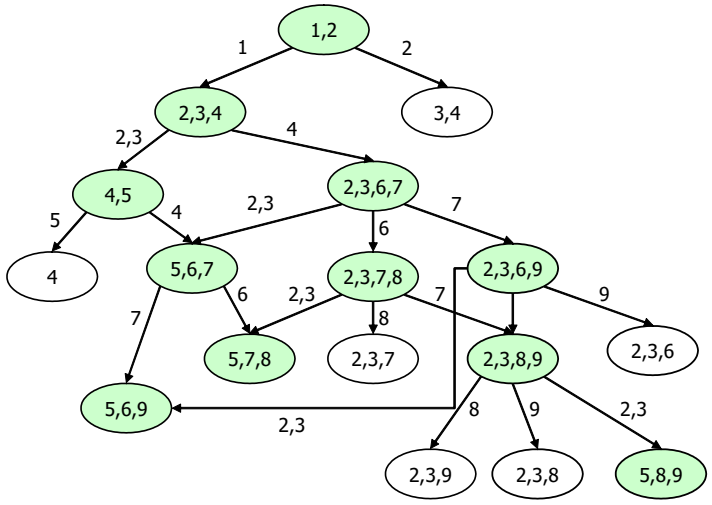

Figure 4.4: Hierarchy of Cutsets (not a complete tree).

The preceding illustration demonstrates the inapplicability of SP techniques to our problem when the *udc*'s represent the stages. While this might be the intuitive way to identify a "stage" in our problem, there could very well be other ways to model the problem and apply SP to solve it. We however conjecture that enforcing nonanticipatory constraints in a sequential decision process such as ours is not possible.

4.4 Approach Via Simulation-Cum Optimization

As seen from the discussion in the preceding section, our problem is not amenable to stochastic programming techniques. Therefore there is need to develop a mathematical framework that can provide some useful information for project planning. One concept that comes to the forefront is obtaining a "good" estimate of project metrics such as (a) time to completion, (b) total cost to completion, and (c) probability of meeting a given due date. We attempt to address these three problems using a simulation-cum optimization approach.

Our model essentially solves an integer program (described in Section 3.2 on page 29) corresponding to each realization in a replication. This leads us to two pertinent issues, namely:

- Minimize the number of samples required to represent the joint distribution of the random variables as accurately as possible,
- Minimize the variance of the desired sample mean response (total project cost, project completion time).

4.4.1 Variance Reduction Techniques

In order to estimate the expected response of a single simulated system one often uses correlation-induction methods such as antithetic variates (AV) and Latin hypercube sampling (LHS) to obtain negatively correlated replications of the response, thereby reducing the variance of the sample mean response.

A useful condition that often guarantees negative induced correlation is based on the notion of *negative quadrant dependence* defined by Lehmann [1966]. A bivariate random vector $(A_1, A_2)^T$ is negatively quadrant dependent if

$$\Pr\{A_1 \leq a_1,\ A_2 \leq a_2\} \leq \Pr\{A_1 \leq a_1\} \cdot \Pr\{A_2 \leq a_2\},\ \forall\, a_1, a_2$$

Avramidis and Wilson [1996] have conducted a detailed study on the implementation of integrated variance reduction techniques to simulation of stochastic activity networks. For the sake of demonstrating the effectiveness of these techniques, they consider a project represented as a directed acyclic graph, $G = (N, A)$, with the objective of estimating the

distribution of the time to realize the sink node — that is, the time to complete the project while respecting all precedence constraints.

In our problem, the system input comprises the work content realizations, and the system response is the output from an optimization process. We are primarily concerned with estimating the total project cost and the time to completion. Of secondary interest is the distribution of resource consumption per period.

We now outline the Latin hypercube sampling method and its properties. This method is due to McKay et al. [1979].

4.4.1.1 Latin Hypercube Sampling (LHS)

Consider for example our problem, where the random vector $\mathbf{W} = (W_1, \ldots, W_n)$ represents the work content of the n activities in the project that need to be sampled from a given distribution $F(\cdot)$. Assume further that we need to conduct k replications. We can generate these k correlated replications via Latin hypercube sampling as follows:

- Let $\mathbf{P} = (p_{ij})$ be a $k \times n$ matrix, where each column of P is an independent random permutation of the integers $\{1, 2, \ldots, k\}$.
- Let ξ_{ij} $(i = 1, \ldots, k;\ j = 1, \ldots, n)$ be nk independent and identically distributed (iid) $U(0, 1)$ random variables independent of P.
- The input random numbers are generated according to the following relation:

$$U_i^{(j)} = \frac{p_{ij} - 1 + \xi_{ij}}{k} \tag{4.11}$$

Now, let W_{ij} be the ith simulated value. We can transform the probability values sampled into the value W_{ij} using the inverse of the distribution

$$W_{ij} = F_j^{-1}(U_i^{(j)})$$

To get an idea of the stratification that occurs in LHS, consider a simple example — say we have three variables ($n = 3$) and we generate five random samples ($k = 5$) of each using LHS. In simple terms, the LHS method would "draw" a random number from each of the following 5 subintervals: (0, 0.2], (0.2, 0.4], (0.4, 0.6], (0.6, 0.8], and (0.8, 1.0). This can be verified from the resulting output shown in Table 4.2.

We now outline some key properties of LHS.

Table 4.2: Sample Output of LHS: $n = 3$, $k = 5$ (left); Random permutation vectors (right)

		Variable		
		1	2	3
Replication	1	0.1709	0.9835	0.3812
	2	0.6814	0.5657	0.6384
	3	0.5980	0.1431	0.4738
	4	0.3004	0.2921	0.1886
	5	0.8725	0.7123	0.8913

	Variable		
	1	2	3
Permutation	1	5	2
	4	3	4
	3	1	3
	2	2	1
	5	4	5

(a) Let $G_{\text{LH}}^{(k)}$ denote the distribution of each k-dimensional column vector of input random numbers generated in this way, so that

$$\mathcal{U}_j \sim G_{\text{LH}}^{(k)} = [U_j^{(1)}, \ldots, U_j^{(k)}]^T$$

is generated according to (4.11).

(b) For each $j\,(j = 1, \ldots, n)$, the components of the column vector $\mathcal{U}_j$ form a stratified sample of size k from the uniform distribution on the interval (0, 1) such that there is a single observation in each stratum and the observations within the sample are negatively quadrant dependent. Morover, the different stratified samples of size k are independent.

(c) Since each column of $\mathbf{P}$ is a random permutation of the integers $\{1, \ldots, k\}$, the following hold:

- Each element p_{ij} for $i = 1, \ldots, k$ has the discrete uniform distribution on the set $\{1, 2, \ldots, k\}$. Therefore, by definition of (4.11) the variate p_{ij} randomly indexes a subinterval of the form $((l-1)/k, l/k]$ for some $l \in \{1, \ldots, k\}$.
- Every subinterval of the form $((l-1)/k, l/k]$ for $l = 1, \ldots, k$ contains exactly one of the negatively quadrant dependent random numbers $\{U_i^{(j)} : i = 1, \ldots, k\}$, implying that the components of $\mathcal{U}_j$ constitute a stratified sample of the uniform distribution on (0, 1).

(d) Since ξ_{ij} is a random number sampled independently of p_{ij}, $U_i^{(j)}$ is uniformly distributed in the subinterval indexed by p_{ij}; and in turn $U_i^{(j)}$ is uniformly distributed in

(0, 1).

(e) Finally the column vectors $\mathcal{U}_1, \ldots, \mathcal{U}_n$ are independent because the random permutations that constitute the columns of $\mathbf{P}$ and the random numbers $\{\xi_{ij} : i = 1, \ldots, k;\ j = 1, \ldots, n\}$ are all generated independently.

4.4.2 Numerical Example

Consider the project defined by the graph in Fig. 4.3. The *PERT estimates* of the activity work contents are defined in Table 4.3. The corresponding parameters of the Beta distribution are also indicated in the last two columns.

Table 4.3: Data for Example Network 3

PERT Estimates			Beta dist.	
min (a)	mode (m)	max (b)	α	β
2	5	8	4	4
1	4	25	1.48	4.36
4	10	16	4	4
1	4	13	2.2	4.6
3	9	15	4	4
2	5	14	2.2	4.6
1	4	13	2.2	4.6
1	2.5	7	2.2	4.6
2.5	4	11.5	1.69	4.46

The parameters for the Beta distribution can be derived (Law and Kelton [2000]) as follows:

$$\alpha = \frac{4 + 3\left(\frac{b-m}{m-a}\right) + \left(\frac{b-m}{m-a}\right)^2}{1 + \left(\frac{b-m}{m-a}\right)^2} \tag{4.12}$$

$$\beta = \frac{1 + 3\left(\frac{b-m}{m-a}\right) + 4\left(\frac{b-m}{m-a}\right)^2}{1 + \left(\frac{b-m}{m-a}\right)^2} \tag{4.13}$$

Other project data are as follows: upper limit on resource availability $R = 4$, project due date $T_s = 25$, cost of tardiness = 10, and resource modes {0.5, 1, 2}.

In addition to the beta distribution, two alternative distributions were considered for the activity work content, W_j, $j = 1, \ldots, 7$:

- W_j has a uniform distribution in the interval (a_j, b_j).
- W_j has a normal distribution in the interval (a_j, b_j), and with mean $\mu_j = 0.5(a_j + b_j)$ and variance $\sigma^2 = [(b_j - a_j)/6]^2$.

The random numbers and the corresponding random variables (uniform, normal, and beta) under Monte Carlo and LHS methods are generated using MATLAB 6.5. These data are then read in into the AMPL modeling language to be used for the simulation runs. The optimization model is solved using ILOG CPLEX 9.0. For each distribution, 5 runs of 100 replications each were conducted and the following statistical information for the total project cost (averaged over 5 runs) are tabulated in Table 4.4:

- p-deciles D(p) for p = 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, and 0.1
- sample mean and standard deviation (denoted by 'Std Dev' in the table), standard error of the mean (denoted by 'Std Err'), and coefficient of variation (denoted by 'CoV')

Table 4.4: Summary of Results for Project Completion Cost from Monte Carlo (MC) and LHS runs

	Distribution (MC)				Distribution (LHS)		
p-deciles	Uniform	Normal	Beta	p-deciles	Uniform	Normal	Beta
D(0.9)	350	326	223	D(0.9)	366	308	234
D(0.8)	329.4	287	226	D(0.8)	330	290	219
D(0.7)	306	273	210	D(0.7)	312	281.5	209
D(0.6)	289.7	264	200	D(0.6)	292.7	274	197
D(0.5)	277	260	193.5	D(0.5)	275	266	187.5
D(0.4)	264	254	175	D(0.4)	257.3	255.5	182
D(0.3)	251	245.5	166	D(0.3)	228	243.5	172
D(0.2)	228	232	157	D(0.2)	208	207	163
D(0.1)	201.4	220	148	D(0.1)	185	213	144
Mean	274	263.41	190.29	Mean	278	262.52	190.57
Std Dev	69.11	42.21	35.12	Std Dev	59.75	35.82	34.27
Std Err	7.01	4.22	3.51	Std Err	6.00	3.78	3.42
CoV	25.20	16.02	18.45	CoV	21.45	14.40	17.98

Table 4.5: P-deciles for the Resource Allocation Vector (Uniform (LHS) Case)

	Activity								
p-deciles	1	2	3	4	5	6	7	8	9
D(0.9)	2	1	1	2	2	1	1.5	1	1.5
D(0.8)	2	1	1	2	2	1	1	1	1
D(0.7)	2	1	1	2	2	1	1	1	1
D(0.6)	2	1	1	2	1	1	1	1	1
D(0.5)	2	1	1	2	1	1	1	1	1
D(0.4)	2	1	1	1	1	1	1	0.5	1
D(0.3)	1	0.5	1	1	1	1	1	0.5	1
D(0.2)	1	0.5	0.5	1	1	0.5	0.5	0.5	0.75
D(0.1)	1	0.5	0.5	1	1	0.5	0.5	0.5	0.5

Observe that in case of uniform and normal distributions, the reduction in variance achieved by using LHS over simple Monte Carlo sampling is approximately 25% and 28%, respectively. In the case of beta distribution, however, the reduction is only 5%.

Note that the output from the simulation run does not provide the user with an "optimal resource allocation vector." Instead what one would find useful are the p-deciles of resource allocation for each activity as displayed in Table 4.5. This would in some sense provide a measure of utilization of resources for each activity. Similar tabulation can be made for the total resource usage of resources across all activities per period.

Based on the data available in Table 4.4 we are now in a position to address the following related problems of interest.

Problem 1. Given resource limit value R and confidence probability p, determine the value of the total project cost that has a probability of at least p.

The solution to this problem can be obtained from the statistical data in Table 4.4 as follows:

(a) Determine integers q and $q+1$, $0 \leq q \leq 9$, satisfying $q/10 \leq p \leq (q+1)/10$.

(b) Calculate the value of total cost

$$z_p = D(q/10) + \{D((q+1)/10) - D(q/10)\}(10p - q)$$

Say for example, one needs to solve the problem for values of $R = 4$ and confidence probability $p = 0.75$. The solution for the case where the work content of activities are independent and uniformly distributed within the specified ranges (and generated via LHS) can be obtained as follows:

$$z_{0.75} = D(0.7) + \{D(0.8) - D(0.7)\} \cdot (7.5 - 7) = 312 + \{330 - 312\} \cdot 0.5 = 321$$

Problem 2. Given resource limit value R and available budget, determine the confidence probability p of not exceeding the budget.

The solution to this problem is similar to that of Problem 1 and is based on using linear interpolation techniques. Say for example, one needs to solve the problem for $R = 4$ and

an available budget of 300. The solution for the case where the work content of activities are independent and uniformly distributed within the specified ranges (and generated via LHS) can be obtained by solving the following:

$$300 = D(0.6) + \{D(0.7) - D(0.6)\} \cdot (10p - 6)$$

The desired confidence probability $p = 0.64$.

Problem 3. Determine the minimal value R that ensures that the available budget is not exceeded with confidence probability p.

In order to illustrate the solution of this problem, consider the network displayed in Fig. 4.5.

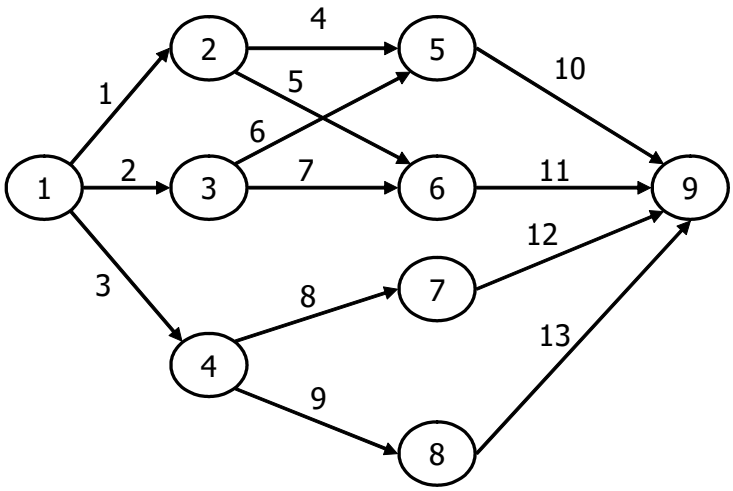

Figure 4.5: Example Network 4 (AoA).

We use this network because it has more activities that can be executed in parallel than the example used in the previous two cases, thereby enabling us to easily observe the benefit of increasing resource availability. The work content of the activities is defined as

$$W_j \sim \text{Uniform}[a_j, b_j], \text{ for } j = 1, \ldots, 13.$$

where $\{a_j, b_j\}$ are defined in Table 4.6.

Table 4.6: Lower and Upper Bounds for W_j

j	1	2	3	4	5	6	7	8	9	10	11	12	13
a_j	2	1	4	1	3	2	1	1	2.5	3	5	2	4
b_j	8	25	16	13	15	14	13	7	11.5	8	12	15	8

They are generated using LHS. Other problem data are: resource cost $c_r = 1$, tardiness cost $c_t = 10$, due date $T_s = 10$. After conducting the simulation experiments for varying values of R, we tabulate the p-decile information corresponding to the total project cost in Table 4.7. Depending on the budget availability and the resource modes available, one can expand the table to include p-deciles for more values of R. From such a table, one can obtain the minimum value of R needed to stay within the budget with the desired confidence probability.

Table 4.7: P-deciles of Total Cost for Varying Values of R (R = 5, 6, 7, 8, ∞)

	$R = 5$	$R = 6$	$R = 7$	$R = 8$	$R = \infty$
D(0.9)	325.25	314.75	314.75	304	304
D(0.8)	305.25	293	293	287.75	287.75
D(0.7)	292.5	278.25	278.25	274.75	274.25
D(0.6)	280.25	268	268	262.5	262.25
D(0.5)	269.5	259	259	252.75	252
D(0.4)	254.25	251.25	251.25	245.25	244.75
D(0.3)	243.25	234	234	226.25	226.25
D(0.2)	232	222	222	217.75	217.75
D(0.1)	206.25	202.5	202.5	194.5	193

Chapter 5

Conclusions and Future Research

The objective of this research is to provide a formal and rigorous treatment to the problem of optimally allocating resources in project networks under both deterministic and stochastic conditions. We introduce the concept of *work content* of an activity, and develop mathematical models to solve the problem when the work content is deterministically known, or known only in probability. This chapter summarizes the work done so far and proposes the directions for future research.

5.1 Main Conclusions of the Research

The classical Resource-Constrained Project Scheduling Problem (RCPSP) deals with the scheduling of precedence- and resource-constrained activities in order to minimize the project completion time. Our problem focusses on how to optimally allocate resources that are limited in availability to the precedence-constrained activities such that the total cost of resources and tardiness is minimized. Our approach to the problem is fundamentally different in perspective from previous treatments of the “classical" RCPSP. In our model, “internal" uncertainty resides in the estimate of *work content*, the decision is the resource allocated to each activity, and the duration of an activity is derived from its work content and the amount of resource allocated to it. In Chapter 2 we reviewed the literature on the RCPSP and its many variants. We observe that the best known approaches to solve the RCPSP still rely on dynamic programming or branch-and-bound. For the stochastic case,

there has been very little literature published so far. Those available focus on scheduling and deal with either minimization of *expected* makespan, or minimization of weighted tardiness, or estimation of the probability of meeting a specific project due date.

In Chapter 3, as a prelude to the analysis under stochastic conditions, we address the problem assuming deterministic knowledge of the activity's *work content*. We develop a nonlinear programming (NLP) model to handle the commonly encountered situations in practice. We start with the simplest scenario where unlimited resources are available and then consider individual resource constraints on activities before proceeding to include additional aggregate resource constraints on activities in *parallel*. The NLP model in the case of aggregate constraints on resource availability is actually an *integer*-NLP which expands in size dramatically as the number of parallel paths increase, and consequently its solution becomes more and more intractable, given the fact that nonlinear programming solvers are not as advanced in integer-variables-handling capabilities as their linear counterparts. We also alert the reader to another major limitation of the NLP model — currently our formulation cannot handle aggregate constraints when applied to non–series/parallel graphs. We also develop an integer programming (IP) model to overcome the limitations of the NLP model regarding inapplicability to non-series/parallel graphs. While it does have the usual limitations in terms of nonlinear growth in problem size with increase in problem parameters (recall that integer linear programming (ILP) is $\mathcal{NP}$-complete), computational experiments show that it tends to perform well on problems with up to 50 activities.

In Chapter 4 we treat the problem assuming uncertainty in activity *work content*. We first assume the *work content* of activities to be independent exponential random variables. The problem as it is currently addressed by Tereso et al. [2001], assumes the resource allocation to an activity, once made, remains invariant and cannot be changed for the duration of the activity. Recognizing that this is at variance with common practice where the manager does indeed change the resource allocation dynamically according to changes in the state of the project, we model our problem as a Continuous-Time Markov Chain (CTMC) and propose a policy iteration-like approach based on the state space of the CTMC. Its drawback is the vast increase in the state space with increase in the size of the project network coupled with the exponential increase in the solution space.

We then provide an overview of stochastic programming (SP) and investigate its applicability to our problem. To aid the understanding of the intricacy of our problem, we discuss a recently researched stochastic project scheduling problem that uses the concept of stochastic programming. We provide a simple example to illustrate the fact that standard stochastic programming formulation cannot be applied to our problem. Finally we outline a simulation-based optimization approach. This approach incorporates efficient sampling techniques such as Latin hypercube sampling (LHS) to achieve variance reduction with smaller sample sizes. The optimization model used is the IP model developed in Chapter 3. We wish to emphasize that this approach should be used only for estimation purposes — that is, as a tool to aid budgeting, bidding, etc. Unlike in the deterministic case, the output from the simulation does not provide a one-to-one mapping of the solution to the objective value. Instead, what one obtains is a set of objective function values from which one can obtain the mean, variance, percentiles, etc.

As alluded to earlier, the formulation enables us to answer three relevant problems: (1) Given resource limit value R and confidence probability p, determine the value of the total project cost that has a probability of at least p. (2) Given resource limit value R and available budget, determine the confidence probability p of not exceeding the budget. (3) Given the available budget and confidence probability p, determine the minimal value R that is needed to meet the deadline.

5.2 Directions for Future Research

We now outline the challenges ahead in the future course of research:

- Develop a mathematical model to handle the case of general precedence. One needs to carefully study the currently available method by Kamburowski et al. [2000] that has the following features: (*i*) it is applicable only to the AoA mode of representation (note that this need not necessarily be the mode in which the project is presented), and, most importantly, (*ii*) it relies on *node reduction* ,i.e., elimination of arcs (or activities in our context). One needs to develop a methodology that is able to transform a non–series/parallel graph to a series/-parallel one and subsequently reconcile the optimal solution of the transformed

graph with that of the original graph.

- The IP model developed in this thesis can be easily extended to the case where more than one type of renewable resource is available. However, it can pose a computational burden very easily. From a practical standpoint, there is strong motivation for developing clever bounding methods and their use in conjunction with heuristics such as Tabu search.
- A more detailed study of variance reduction techniques is needed from the perspective of application to cases where the input random variables need to be generated using acceptance-rejection techniques.
- Investigate the possibility of stochastic global optimization via meta-heuristics, especially the *Electromagnetism-like algorithm*, and its validation via Monte Carlo based simulation.

Bibliography

M. K. Agrawal, S. E. Elmaghraby, and W. S. Herroelen. DAGEN: A generator of testsets for project activity nets. *Euro. J. Operl Res.*, 90:376–382, 1996.

A. N. Avramidis and J. R. Wilson. Integrated variance reduction strategies for simulation. *Operations Research*, 44(2):327–346, 1996.

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

S. I. Birbil and S-C. Fang. An electromagnetism-like mechanism for global optimization. *Journal of Global Optimization*, 25:263–282, 2003.

J. Czyzyk, M. Mesnier, and J. Moré. The NEOS Server. *IEEE Journal on Computational Science and Engineering*, 5:68–75, 1998.

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

E. Demeulemeester, B. DeReyck, and W. Herroelen. The discrete time/resource trade-off problem in project networks - a branch-and-bound approach. Technical report, No. 9717, Department of Applied Economics, K. U. Leuven., 1997.

B. DeReyck and W.S. Herroelen. On the use of the complexity index as a measure of complexity in activity networks, res. report no. 9332. Technical report, Department of Applied Economics, Katholieke Universiteit Leuven, Debériotstraat 36, B-3000 Leuven, Belgium, 1993.

E. Dolan. The NEOS Server 4.0 administrative guide, technical memorandum anl/mcs-tm-250. Technical report, Mathematics and Computer Science Division, Argonne National Laboratory, 2001.

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

A. A. Fernandez and R. L. Armacost. The role of the nonanticipativity constraints in commercial software for the stochastic project scheduling. *Computer and Industrial Engineering*, 31(1/2):233–236, 1996.

R. Fletcher and S. Leyffer. Numerical experience with lower bounds for miqp branch-and-bound. *SIAM J. Optimization*, 8(2):604–616, 1998.

D. R. Fulkerson. A network flow computation for project curves. *Management science*, 7: 167–178, 1961.

M. R Garey and D. S. Johnson. *Computers and Intractibility - A Guide to the Theory of NP-Completeness*. W. H. Freeman & Company, San Fransisco, 1979.

M. R. Garey, R. L. Graham, R. L. Johnson, and C. Yao. Resource–constrained scheduling as genaralized bin packing. *Journal of Combinatorial Theory*, (A)21:257–298, 1976.

D. Golenko-Ginzburg and A. Gonik. Stochastic network project scheduling with non-consummable limited resources. *International Journal of Production Economics*, 48:29–37, 1997.

W. Gropp and J. Moré. Optimization environments and the NEOS Server. *Approximation Theory and Optimization, M. D. Buhmann and A. Iserles, eds., Cambridge University Press*, pages 167–182, 1997.

W. J. Gutjahr, C. Strauss, and E. Wagner. A stochastic branch-and-bound approach to activity crashing in project management. *INFORMS Journal on Computing*, 12(2):125–135, 2000.

S. Hartmann. A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics*, 45:733–750, 1998.

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

T. J. Hindelang and J. F. Muth. A dynamic programming algorithm for decision CPM networks. *Operations Research*, 27:225–241, 1997.

J. Kamburowski, D. J. Michael, and M. F. M. Stallmann. Minimizing the complexity of an activity network. *Networks*, 36(1):47–52, 2000.

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

V. G. Kulkarni and V. G. Adlakha. Markov and Markov-Regenerative PERT networks. *Operations Research*, 34:769–781, 1986.

A. M. Law and W. D. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, Boston, Third edition, 2000.

E. L. Lehmann. Some concepts of dependence. *Anns. Math Statist.*, 37:1137–1153, 1966.

S. Leyffer. Integrating sqp and branch-and-bound for mixed integer nonlinear programming. *Computational Optimization and Applications*, 18:295–309, 2001.

M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21:239–245, 1979.

M. F. Neuts. *Structured Stochastic Matrices of M/G/1 type and their Applications*. Probability:Pure and Applied - A Series of Testbooks and Reference Books, 1989.

J. H. Patterson. Exact and heuristic solution procedures for the constrained resource project scheduling problem, vol.i-iii. Technical report, School of Bus. Adm., Indiana University, Bloomington, IN 47401, 1982.

A. Sprecher and A. Drexl. Solving multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm. *European Journal of Operational Research*, 107:431–450, 1998.

F. Stork. *Stochastic Resource-Constrained Project Scheduling*. PhD thesis, Technical University of Berlin , School of Mathematics and Natural Sciences, 2001.

A. P. Tereso, M. M. Araújo, and S. E. Elmaghraby. Adaptive resource allocation in multimodal activity networks. *IJPE*, 92:1–10, 2001.

A. P. Tereso, M. M. Araújo, and S. E. Elmaghraby. Experimental results of an adaptive resource allocation technique to stochastic multimodal projects. Technical report, Universidade do Minho, Guimarães, Portugal., 2003a.

A. P. Tereso, M. M. Araújo, and S. E. Elmaghraby. Basic approximations to an adaptive resource allocation technique to stochastic multimodal projects. Technical report, Universidade do Minho, Guimarães, Portugal., 2003b.

A. P. Tereso, M. M. Araújo, and S. E. Elmaghraby. The optimal resource allocation in stochastic activity networks via the electromagnetism approach. In *Ninth International Workshop on Project Management and Scheduling(PMS '04), Nancy, France*, 2004.

V. Valls, M. Laguna, P. Lino, A. Perez, and S. Quintanilla. Project scheduling with stochastic activity interruptions. *Project Scheduling: Recent Models, Algorithms and Applications, Kluwer Academic Publishers*, pages 333–353, 1998.

R. D. Wollmer. Critical path planning under uncertainty. *Mathematical Programming Study*, 25:164–171, 1985.

Appendix A

Generation of Test Networks

The need for a standard set of test problems has been recognized quite early by both practitioners and researchers. None was available until November 1982 when Patterson [1982] assembled 110 problems of varying degrees of "difficulty". This set became the *de facto* standard testset among researchers in the field. Approximately ten years passed before a group of researchers at Kiel University, led by Kolisch et al. [1995], offered a software that generates nets of varying degrees of "difficulty".

For our computational experiments however, we use the software DAGEN, developed by Agrawal et al. [1996]. This is because their motivation to generate networks was driven by the fact that neither of these prior testsets took cognizance of the more recent result of Bein et al. [1992]. Briefly, Bein et al. [1992] established the "reduction complexity" or "complexity index" (CI), as a measure of non-conformity of a graph to the series/parallel topology. The CI is an important parameter that adds one more dimension to the measure of a graph's "difficulty," and has been confirmed by the work of Elmaghraby [1993] and DeReyck and Herroelen [1993].

We now briefly describe the methodology used in DAGEN to generate source-terminal directed acyclic graphs of given complexity index (CI). The method is based on a three-step approach. The first step generates the so-called "skeleton network" of the specified CI. In the second step, modules of extra nodes and arcs are inserted sequentially in either the skeleton or the current network such that the complexity index of the resulting network remains unaltered. Finally, the values of the various parameters (duration, renewable resource(s), and the non-renewable resource) are added.

For our requirement, we just use the data available after the first two steps of the procedure, that is — a graph with the desired number of nodes, activities, and complexity index. The details of these two steps are discussed in the following sections.

A.1 Initial Framework

A *skeleton* is a network of given complexity that employs a minimal number of nodes and arcs. The skeleton can be generated in many ways. In this program we propose a simple linear structure for the skeleton. The generation of a skeleton proceeds as follows.

Given the complexity index (CI) = c, a skeleton with the minimum number of nodes and arcs is a chain of $c + 3$ nodes (and $c + 2$ arcs). The nodes are numbered 1 to n (where $n = c + 3$). Node 1 is treated as the source node and node n as the terminal node.

To obtain the skeleton, node 1 is connected to nodes 2 to $n - 1$; nodes 2 to $n - 2$ are connected to node n and also to the node with the immediate higher number (i.e., 2 to 3, 3 to 4, etc.). Finally node $n - 1$ is connected to node n.

A.2 The Generation of the Network

In the second step of the procedure, modules consisting of at most one node and at most three arcs are inserted into the current network such that the CI of the resulting network remains unaltered. The modules represent the basic building blocks of the network. A total of five modules were considered for this purpose, as described next:

(a) **Series Expansion**: If an arc (i, j) already exists between two randomly selected nodes i and j, and $i \prec j$, then a node k can be inserted between nodes i and j such that the original arc (i, j) is *replaced* by two arcs (i, k) and (k, j). This results in a net addition of one node and one arc.

(b) **Parallel Expansion**: If an arc (i, j) exists between two randomly selected nodes i and j, and $i \prec j$, then a node k can be inserted by adding two arcs (i, k) and (k, j). This results in a net addition of one node and two arcs.
Note: Arc (i, j) is not deleted here.

(c) **Transitive Direct Expansion:** If the randomly selected nodes i and j are not connected by a direct arc then an arc may be inserted between nodes i and j. The insertion of such an arc is permissible if $i \prec j$ and the complexity of the resulting network — that is, after inserting the arc — remains unaltered. This results in a net addition of one arc and zero nodes.

(d) **Transitive Series Expansion:** If the randomly selected nodes i and j are not connected by a direct arc then a node k and two arcs (i, k) and (k, j) may be inserted between nodes i and j. The insertion of this module is permissible if $i \prec j$ and the complexity of the resulting network — that is, after inserting the arc — remains unaltered. This results in a net addition of one node and two arcs.

(e) **Transitive Parallel Expansion:** If the randomly selected nodes i and j are not connected by a direct arc then a node k and three arcs (i, j), (i, k), and (k, j) may be inserted between nodes i and j. The insertion of this module is permissible if $i \prec j$ and the complexity of the resulting network — that is, after inserting the arc — remains unaltered. This results in a net addition of one node and three arcs.

Appendix B

Listing of AMPL Code

B.1 Nonlinear Programming Model

```
#Example project in Chapter 3
#Aggregate constraints

param m := 10;    # Number of activities
param n := 7 ;    # Number of nodes
param k := 2 ;    # Number of resources
param eps := 0.5;
param due := 36;
param bigM := 100;
param R1 := 8;
param R2 := 8;

set M := 1..m;
set N := 1..n;
set K := 1..k;
set NxN := {N cross N};
set MxK := {M cross K};
param w_res1{M}>=0;
param w_res2{M}>=0;

var v >=0;
var x2{M} >=1,<=3;
var x{M} >=1, <=3;          # Amount of resource allocated to activity m
var t{N} >=0;               # Realization of node n
var y{M} >=0;
var y2{M} >=0;
var delta {NxN} binary;
var gamma {NxN} binary;

# Objective function
```

```
minimize total_cost:
150*(x[1]^0.6) + 382.09*(x[1]^0.9947) +
186*(x[2]^0.6) + 470.01*(x[2]^0.9947) +
228*(x[3]^0.6) +  55.90*(x[3]^0.9947) +
 96*(x[4]^0.6) + 300.56*(x[4]^0.9947) +
138*(x[5]^0.6) + 238.18*(x[5]^0.9947) +
 96*(x[6]^0.6) + 226.90*(x[6]^0.9947) +
 90*(x[7]^0.6) +  62.46*(x[7]^0.9947) +
174*(x[8]^0.6) + 304.53*(x[8]^0.9947) +
 48*(x[9]^0.6) + 150.28*(x[9]^0.9947) +
 84*(x[10]^0.6) + 190.11*(x[10]^0.9947) +
 <<due,(due+5);0,200,800>>t[7];

subject to
ind_dur1{j in M}: y[j] >= w_res1[j]/x[j]^0.9;
res2{j in M}:x2[j] = (w_res2[j]/w_res1[j])^(1/0.95)*(x[j]^(0.9/0.95));
ind_dur2{j in M}:y2[j] = w_res2[j]/x2[j]^0.95;

# General project completion constraints
con24:t[1] = 0;
con13a:t[2] = t[1] + y[1];
con14a:t[3] = t[2] + y[2];
con15:t[4] >= t[2] + y[3];
con16:t[5] >= t[3] + y[4];
con16a:t[5] >= t[3] + y[5];
con17:t[5] >= t[4] + y[6];
con17a:t[5] >= t[4] + y[7];
con18:t[6] >= t[5] + y[8];
con18a:t[6] >= t[5] + y[9];
con19:t[7] >= t[6] + y[10];

   # Set of 9 constraints to handle aggregate resource constraints
conZ:x[2] + x[3] <=R1;
conZ1:x2[2] + x2[3] <=R2;
conY:x[8]+x[9] <=R1;
conY1:x2[8]+x2[9] <= R2;
con1:t[3] - t[4] <= -eps + bigM*delta[3,4];
con2:t[4] - t[3] <= -eps + bigM*gamma[3,4];
con3:t[3] - t[4] <= -eps + bigM*(1-delta[3,4]+gamma[3,4]);
con4:t[4] - t[3] <= -eps + bigM*(1+delta[3,4]-gamma[3,4]);
con5:t[3] - t[4] <= eps + bigM*(2-delta[3,4]-gamma[3,4]);
con6:t[4] - t[3] <= eps + bigM*(2-delta[3,4]-gamma[3,4]);

con7:x[3] + x[4] + x[5] <= R1 + bigM*(1 - delta[3,4] + gamma[3,4]);
con8:0.4288*x[3]^0.9474 + 1.5324*x[4]^0.9474 + 1.1378*x[5]^0.9474 <=
R2 + bigM*(1 - delta[3,4] + gamma[3,4]);

con9:x[2] + x[6] + x[7] <= R1 + bigM*(1 + delta[3,4] - gamma[3,4]);
```

```
con10:1.3767*x[2]^0.9474 + 1.3314*x[6]^0.9474 + 0.7215*x[7]^0.9474 <=
R2 + bigM*(1 + delta[3,4] - gamma[3,4]);

con11:x[4] + x[5] + x[6] + x[7] <= R1 + bigM*(2 - delta[3,4] - gamma[3,4]);
con12:1.5324*x[4]^0.9474 + 1.1378*x[5]^0.9474 + 1.3314*x[6]^0.9474 +
      0.7215*x[7]^0.9474 <= R2 + bigM*(2 - delta[3,4] - gamma[3,4]);

# Problem data
data;
param :     w_res1  w_res2:=
1   25   34
2   32   42
3   38   17
4   16   24
5   23   26
6   16   21
7   15   11
8   29   33
9    8   12
10  14   18;
```

B.2 Integer Programming Model

B.2.1 AMPL model file

```
param due  >= 0;
param C_T  >= 0;
param T >= 0;
param R{1..T} >= 0;
set NODE;
set ARC within (NODE cross NODE);
set ACTIVITY;

param nModes integer > 0;
param x {1..nModes} >= 0;
param cost {ACTIVITY} >= 0;
param ac {ARC} in ACTIVITY;
param Work {ACTIVITY} >= 0;

var v >= 0;
var st {ACTIVITY} >= 0 integer;
var fi {ACTIVITY} >= 0 integer;
var tr {NODE} >= 0 integer;
var z {ACTIVITY, 1..nModes} binary;
var alfa {ACTIVITY, 1..nModes,1..T} binary;
var beta {ACTIVITY, 1..T} binary;
var delta {ACTIVITY, 1..T} binary;

# Obj function - Minimize sum of resource costs & tardiness cost #
minimize TCost:
sum {(i,j) in ARC, k in 1..nModes} z[ac[i,j],k]*x[k]*cost[ac[i,j]] + <<due;0,C_T>>v;

s.t. C1 {(i,j) in ARC}:
st[ac[i,j]] >= tr[i];

# Define finish time of activity (i,j) #
s.t. C2 {(i, j) in ARC}:
fi[ac[i,j]] >= st[ac[i,j]] + sum {k in 1..nModes} z[ac[i,j],k]*(Work[ac[i,j]]/x[k]);

# Realization of node j is the max of all activities terminating at that node#
s.t. C3{(i,j) in ARC}:
tr[j] >=  fi[ac[i,j]];

# Each activity is executed exactly in 1 mode #
s.t. C4 {(i,j) in ARC}:
sum{k in 1..nModes}z[ac[i,j],k] = 1;

# Project completion time #
```

```
s.t. C5{(i,j) in ARC}:
v >= fi[ac[i,j]] ;

# ********** Constraints for indicator variable, a[j,k,t] ******** #

s.t. C6{(i,j) in ARC, t in 1..T}:
sum{k in 1..nModes}alfa[ac[i,j],k,t] <= 1 + (t - st[ac[i,j]])/T;

s.t. C7{(i,j) in ARC, t in 1..T}:
sum{k in 1..nModes}alfa[ac[i,j],k,t] <= 1 + (fi[ac[i,j]] - t )/T;

s.t. C8{(i,j) in ARC, t in 1..T}:
beta[ac[i,j],t] >= (t - st[ac[i,j]])/T;

s.t. C9{(i,j) in ARC, t in 1..T}:
delta[ac[i,j],t] >= (fi[ac[i,j]] - t + 1)/T;

s.t. C10{(i,j) in ARC, t in 1..T}:
sum{k in 1..nModes}alfa[ac[i,j],k,t] >= beta[ac[i,j],t] + delta[ac[i,j],t] - 1;

# ********************************************************************* #
s.t. C11{(i,j) in ARC, k in 1..nModes,t in 1..T}:
alfa[ac[i,j],k,t] <= z[ac[i,j],k];

# Resource constraints #
s.t. C12{t in 1..T}:
sum{(i,j) in ARC, k in 1..nModes}alfa[ac[i,j],k,t]*x[k] <= R[t];
```

B.2.2 AMPL data file

```
# AMPL data file for 20 activities

set NODE := 1 2 3 4 5 6 7 8 9 10;

set ACTIVITY := 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20;

param R default 5;
param T      := 30;
param nModes := 3;
param x := 1 1 2 2 3 3;
```

```
param due := 10;
param C_T := 10;

set ARC :=
1 2
1 4
1 5
1 6
1 7
1 9
2 3
3 4
3 10
4 5
4 10
5 6
5 10
6 7
6 10
7 8
7 9
7 10
8 9
9 10 ;

param ac :=
1 2 1
1 4 2
1 5 3
1 6 4
1 7 5
1 9 6
2 3 7
3 4 8
3 10 9
4 5 10
4 10 11
5 6 12
5 10 13
6 7 14
6 10 15
7 8 16
7 9 17
7 10 18
8 9 19
9 10 20 ;
```