# A Constraint Based Approach Towards Resource Management

Mahmudul Gani

TR-EIL-97-2

Enterprise Integration Laboratory
Department of Mechanical & Industrial Engineering
University of Toronto
4 Taddle Creek Road
Toronto, ON M5S 3G9

Tel: +1(416)978-6823
Fax: +1(416)971-2479
Internet: eil@ie.utoronto.ca

# A Constraint Based Approach Towards Resource Management

by

Mahmudul Gani

A project report submitted in conformity with the requirements
for the degree of Master of Engineering
Department of Mechanical and Industrial Engineering
University of Toronto

September, 1997

# Acknowledgments

There have been numerous people who have given me their encouragement, their advice and their assistance in all possible ways during the period of this research. I am extremely grateful to all those who have helped me. To mention them all here would make too long a list.

With special appreciation, I want to thank Prof. Mark Fox for his guidance, valuable suggestions and never-ceasing encouragement in my research work. Further I would like to thank my other colleagues at the Enterprise Integration Laboratory. A part of this research has been carried out at the R &D section of Numetrix Ltd. I wish to express my sincere gratitude to all my colleagues at Numetrix, specially Chris Beck for his encouraging support and guidance on a continuous basis.

I also acknowledge the grants and scholarships received from the Canadian Commonwealth and Fellowship Plan, Manufacturing Research Corporation of Ontario, IRIS and the University of Toronto during the time of this research.

Finally my warmest thanks are owed to those closest and dearest to me. The most valuable encouragement, support and understanding I have got from my mother Suraya Rahman, my father Atowar Rahman, my wife Aneesa Islam Keya, my brother Anis, my sister Rumi, my brother-in-law Monowar and my parents-in-law. Thank you all.

# Abstract

A Constraint Based Approach Towards
Resource Management

by

Mahmudul Gani

A project report submitted in conformity with the requirements for the degree of
Master of Engineering
Department of Mechanical and Industrial Engineering
University of Toronto

In this work we demonstrated the application of constraint based technique towards solving a broader class of resource management problem. A generic representation and problem solving scheme is presented for solving alternative scheduling problems which involve alternative resources, alternative activities and alternative process plans. A new type of temporal constraint and the associated propagation algorithm are presented. We also introduced the notion of the probability of existence of an activity. A constraint for representing the existence dependency among the activities and the techniques for propagating the probability of existence through these constraints are presented. We developed new heuristics based on the texture measurements of the constraint graph for choosing alternatives during the course of the search. Two sets of experiments were run. In the first set, we demonstrated the validity of our approach in solving alternative jobshop scheduling problem. In the second set of experiment, we demonstrated the applicability of our approach in solving resource management problem.

# Contents

# List of Figures

# List of Tables

# *Chapter 1 Introduction*

This chapter defines the resource management problem and outlines our approach towards representing and solving this problem under the framework of constraint based problem solving approach. It outlines the research objective, research hypothesis and research methodology. The first section provides a definition of the term 'resource management'. Section two provides an overview of the work. The third section provides the motivation of the current research. Section four defines the problem that we intend to solve. The fifth section outlines the objective of the research. Section six discusses the contributions of the present work. The chapter concludes with a note on the report organization.

## 1.1 Introduction

The emergence of the new economic world order provides strong impetus to the organizations to become more competitive. Contrary to the classical belief that the organizations dictate consumer choices, the new economic trends is setting a pace where market demands are shaped by the choices and preferences of the customers. This necessitates the need for the organization to become more agile or responsive to customer needs. An organization which can respond quickly and positively to the ever changing customer demands would succeed in the new global economic scenario.

One of the factor that determines an organization's ability to become more responsive to the customer is its ability to efficiently manage resources across the supply chain. A supply chain encompasses all activities that start "from when the customer first knocks on the door to when the final goods and services are delivered "[Cleaves 96]. Being efficient means that the organization is capable of ensuring that the required resource is available at the right time, at the right quantity and at the right place. The success of an organization lies heavily on its ability to efficiently manage its resources across the supply chain. Resource management has thus emerged as one of the critical function within the enterprise. Recognition of this fact resulted in the search for techniques and tools that can be used by organizations to effectively manage its resources. For many years, resource management has been an active area of research among different research communities. As a result, the term 'resource management'

is used rather loosely in the literature. It's meaning is often context dependent. Thus, we see the use of the term from human resource management to the network resource management to the forest resource management. Before we get into further discussion we need do define the term 'resource management'. [Azarmi 95] provides a definition of this term to be as follows:

"Resource management is concerned with, among other things:

- maintaining an inventory of resources.
- managing supply and installation of new resources
- the allocation and de-allocation of resources
- planning and scheduling of activities and resources."

This covers a wide range of tasks or activities to be executed. Although the definition was posed from the viewpoint of managing telecommunication resources, it is generic enough to cover wide variety of domains. For our present work, we view resource management in the context of manufacturing and our subsequent discussions will be based on the problem of manufacturing resource management.

## 1. 2 Overview of the Work

Reasoning about resources within a manufacturing environment involves deciding about alternatives courses of action. At the planning level, the sources of alternative choices are:

- Means of satisfying customer demand: This involves deciding about how the demand for products can be met. For example, the demand for a certain product can be satisfied in many ways such as, from in-house inventory, or from in-house production, or by an outside supplier/subcontractor, or by a combination of these.
- Allocation of production quantities: Deciding how much production quantity to allocate to different plants.
- Allocation of distribution inventory: This involves deciding about how much inventory to allocate to different distribution channels.
- Modes of transportation: This involves deciding about which mode of transportation to use from a set of alternatives.

- Materials planning: At materials planning level, alternative choices involve deciding over alternative components, alternative raw materials, alternative sub-assemblies etc.
- Purchasing: This involves choosing a supplier from a set of alternative suppliers.

At the scheduling level, the source of alternative choices are primarily due to resource and activity alternatives. The classical scheduling models failed to take a holistic view in terms of deciding about the alternative courses of action. Although work on dealing with alternative resources in scheduling domain are reported in the literature [Nuijten 94], but a comprehensive treatment of alternatives in scheduling is virtually non-existent.

The existence of alternatives increases the complexity of the decision making process associated with the resource management task as it involves trade-off in decision-making. The quality of decision making is greatly influenced by which alternative is chosen. The choice is constrained by the different constraints such as order due date, resource capacity, cost etc. Failure to take into account of these constraints often results in a solution that fails at the execution level. Thus, it is imperative on any resource management system to have a better planning and scheduling tool that would allow an enterprise to make its resource management decision in an integrated manner. By decoupling the task of planning and scheduling the traditional planning and scheduling, tools tend to ignore the interaction effect of these two tasks and failed to give expected results in real life situations. The effect of using these tools for managing resources in a highly dynamic environment results in several unexpected outcomes which includes missed due dates, high inventory accumulation and eventually missed profits. This calls for a need to develop better tools for resource management that is capable of overcoming the shortcomings of the classical approaches.

In the current work, we present an approach that overcomes these shortcomings to a great extent. Our approach involves modeling the resource management problem as a scheduling problem. But in addition to solving the scheduling problem, we also decide about the allocation of quantities to the activities in the process plan i.e., we simultaneously assign quantities to activities and schedule them on the resources. Toward this end, we explored the applicability of constraint based problem solving technique that decides about the alternative courses of action by taking into account both the upstream and downstream constraints of manufacturing planning.

## 1. 3 Motivation of the Current Work

Traditionally, in a manufacturing setting, MRP (Materials Requirement Planning) is used as the materials planning and inventory control tool. Although in the production management literature MRP system is considered to be one of the most successful system for resource management, in real life situations it failed to prove its superiority. This calls for a newer, more innovative approach to deal with the problem of resource management. Moreover, the traditional industrial engineering literature advocates the use of complex mathematical models for inventory management and scheduling. These mathematical models, although in some instances proved to be effective, but fails at least in the following two considerations:

- These models are developed with a very restrictive view in mind and their applicability is limited to a rather restrictive domain. For example, [Chikan 90] in his review provides a list of 336 different inventory models. Some of these models are developed for finite capacity, some for infinite capacity, some for single item, and some for multi-item orders. Some of these allow backorders, others do not. Some considers a fixed lead time while others consider instantaneous delivery. A lack of generality of these models prohibit their use in wide variety of circumstances.
- The another shortcoming of these models is their lack of expressive powers. They fail to capture various real life constraints and assume a more simplistic picture of the problem. This reduces the effectiveness of these models when put into operation in real life contexts. This is particularly true in scheduling domain, which deals with problems that are combinatorial in nature and that involves satisfaction of wide variety of constraints.

Our motivation to work with the resource management problem came from the failure of the existing systems to manage resources in the enterprise in an effective manner. Although the MRP logic of finding product demands using the BOM explosion works well for discrete parts manufacturing, it is the planning and scheduling aspects of MRP that inhibit its effectiveness in real life situations.

The second motivation for this research comes from the success of constraint based approach in the scheduling domain. Recent years have experienced an explosive number of growth in the use of constraint based techniques towards solving scheduling problems. The introduction of AI techniques to scheduling problems is relatively new. Ever since ISIS [Fox 83], the first constraint based scheduling system came into existence, it has remained to be an active area of research in the constraint satisfaction community. After ISIS, the successful applica-

tion of constraint based approach in multitude of domains have been reported in the literature. This includes application in job-shop scheduling to space-shuttle scheduling. The success of these systems motivated us to expand the frontiers of constraint based paradigm to other classes of problems in the manufacturing domain. Much of the work in constraint based approach is confined to the case of job-shop scheduling problems and virtually no work has been done in other areas of manufacturing management such as materials planning and inventory management. This motivated us to explore the potential of applying the constraint based techniques to this virtually unexplored domain of resource management.

## 1.4 Problem Definition

Resource management within the manufacturing context refers to all activities associated with the acquisition, management and distribution of resources within and outside of the enterprise. It encompasses the function of purchasing, inventory management and scheduling of resources. Thus the problem of materials and inventory management, and scheduling falls within the realm of resource management.

Resource management problem, from our viewpoint refers to the problem of planning and allocating resources to satisfy demands by taking into account a varied sets of constraints and the solution of this problem amounts to the satisfaction of these constraints. The constraints that we are interested in are organizational, physical, causal and availability constraints [Fox 1994]. The organization constraints concerns due dates. The physical constraints relates to the functionality of physical resources. Precedence constraints fall under the category of causal constraints. The availability constraints are associated with resource constraints. The interaction of all these constraints together with the simultaneous planning and scheduling aspects of the resource management problem makes it a very difficult problem to solve.

Solving the resource management problem translates into the solving the materials planning and scheduling problem with alternative choices. Our decision considers alternatives of buying, manufacturing/assembling and drawing things from inventory. The problem of resource management has got a very similar look and feel to that of scheduling problems, and we modeled it as an alternative planning and scheduling problem. But, in addition to scheduling of task on resources, we also have to decide which alternative to choose from a set of alternatives.

## 1.5 Objective of the Research

Resource management, within the manufacturing context comprises a myriad of activities that cut across different functional areas of the organization. The objective is to ensure that required resources are available at the right quantity, at right time and at right place. Fulfilling this objective amounts to the satisfaction of varied sets of constraints - both physical and organizational - that are imposed by different functional areas of the organization. This aspect of resource management makes it an ideal candidate for potential application of constraint based problem solving techniques.

The application of the constraint based problem solving technique have demonstrated successful results in solving combinatorial problems such as scheduling. Whereas scheduling problem is also a form of resource management problem, but it fails to acknowledge the complexities associated with higher level resource allocation problem where we have to deal with alternative source of resources to satisfy our demand. Thus, there is a need for new strategies to tackle this problem. The objective of the current research is to explore the applicability of constraint based paradigm to the problem of resource management. We believe that like scheduling domain, the resource management problem can also be modeled as a constraint satisfaction problem and we can explore the application of CSP (Constraint Satisfaction Problem) techniques for this type of problem. More specifically we intend to address the following questions:

- Can we model the resource allocation and management problem as a constrained satisfaction problem?
- Can we overcome the shortcomings of the traditional resource management techniques such as MRP?
- Can we solve this problem using efficient search techniques in reasonable amount of time?

Our objective is *not* to find the optimum solution to these problems. Rather, we are interested in finding ways of modeling and solving these new classes of problems associated with resource management by using the constraint based techniques.

## 1.6 Contributions of the Present Work

The contributions of the current work are six folds:

- Demonstrated the application of constrained based techniques for solving a broader class of resource management problems.
- Proposed and implemented a generic representation scheme for solving alternative scheduling problem that involve alternative resources, alternative activities and alternative process plans.
- Incorporated a new type of temporal constraint and the associated propagation algorithm to facilitate propagation through the alternative activities.
- Introduced the notion of probability of existence of an activity.
- Created propagation techniques to modify an activity's probability of existence by introducing a new type of constraint. This constraint expresses the dependency relationship of activities in a process plan.
- Developed heuristics techniques based on texture measurements of the constraint graph that takes the probability of existence into account while choosing alternatives during the course of the search.

## 1.7 Organization of the Project Report

In Chapter 2, we reviewed the literature relevant to the current work. We present the detailed model for our generic representation of the alternative scheduling problems in Chapter 3. Chapter 4 presents our problem solving paradigm and the texture measurements that are developed for solving this type of problem. Chapter 5 reports the experimental results of our approach on a suite of scheduling problems and validate the applicability of our approach for this kind of problem. Chapter 6 demonstrated how our modeling and problem solving framework can be extended to solve resource management problems. Chapter 7 summarizes the report and provides directions for future work.

# *Chapter 2 Literature Review*

## 2. 1 Introduction

Manufacturing is often referred to as a set of activities that facilitates the conversion of inputs (raw materials) into outputs (useful products). This simplistic definition fails to capture the inherent complexities associated with the process of manufacturing. The essence of manufacturing is well captured by [Plossl 94] in the following words:

> "The essence of manufacturing is flow of materials from suppliers, through plants to customers, and of information to all parties about what was planned, what has happened, and what should happen next"

The decision processes in any manufacturing attempts to answer the following questions [Plossl 94]:

> "1. What is to be made?
> 2. How many, and when are they needed?
> 3. What resources are required to do this?
> 4. Which are already available?
> 5. Which others will be available in time?
> 6. What more will be needed, and when?"

It is obvious from the above questions that, most of the decisions about manufacturing processes revolve around "resources". The management's response towards answering these questions can be captured by the following two tasks:

- determining the quantity of resources and
- allocating these resources to different activities

These decisions are made with the view of satisfying the customer demands in the as effective manner as possible. The two activities that encompasses these two tasks are 'inventory management' and 'scheduling'. Although they seem to deal with two different problems, there are strong interdependencies between them. The decision of one significantly influ-

ences the decision of the other. Failure to understand and take into account such dependencies in developing models of manufacturing results in a poor quality models. In the classical approach these two problems were dealt somewhat independently. The sequential nature of this approach overlooks many of the downstream (execution level) constraints at the planning and scheduling stage. Although there are no dearth of literatures in the areas of scheduling and inventory management most of these works failed to attack the problem at its entirety.

In the absence of a standard terminology that truly integrates these two domains of inventory management and scheduling, we will refer this to be "resource management". Our objective in this research is to explore the applicability of new techniques to these domains in an integrated fashion. Toward this end, we will first provide an overview of the traditional and contemporary approaches for solving the scheduling and inventory management problems.

## 2. 2 Research in Scheduling

Scheduling has been an active area of research in Operations Research (OR) community for the last fifty years. Research in Artificial Intelligence (AI) community took up this problem in early eighties [Fox 83]. Numerous research publications have been reported both in the AI and OR literatures on a wide variety of scheduling problems. For the sake of our discussion, we will broadly classify them into the following two groups:

- non-constraint based scheduling
- constraint-based scheduling.

The objective of our discussion is to compare and contrast these two different research paradigms in scheduling. This is not intended to provide an extensive review of the literature in the respective domains. Several recent articles provided excellent survey of scheduling literatures from both of these paradigms. Please refer to [Graves 81], [Rodammer 88] and [Brown 94] for an extensive review of literatures. Since the current research is based on constraint-based approach, the review in this chapter will primarily be focussed on constraint-based scheduling literature.

## 2. 3 Non-constraint based scheduling methods

Several non-constraint based scheduling methods are reported in the literature. These can be grouped into mathematical programming, bottleneck analysis and dispatch rules. Following sections briefly overview these approaches.

### 2. 3. 1 Mathematical programming

In mathematical programming approach, the scheduling problem is formulated using variables and constraints. The variable represents the decision parameters and the constraints express the relationship among them. Two approaches towards this end are linear and integer programming [Hillier 90]. In linear programming, the variables are continuous and the constraints are described by conjunctive set of linear equations. For optimization problem an objective function is defined whereby the goal of the problem solving is to optimize the function. For non-optimization problem, a dummy optimization function can be used. The difficulty in linear programming involves in mapping of the original problem into set of variables and linear equations. Once the problem is formulated under this framework the problem can be solved using standard LP techniques such as simplex and revised simplex methods.

If the decision variables are not continuous (which is often the case for scheduling) the problem is formulated as integer programming problem. These problems can be solved by branch and bound techniques. Branch and bound is depth-first search which uses an evaluation function to prune off the search space.

Despite the fact that the mathematical programming approach has significantly contributed towards the solution of scheduling problems, the quality of solution is often a function of how the original problem is formulated into variables and constraints [Tsang 95]. The applicability of these techniques is also restricted by the combinatorial explosion of problem space in the presence of large number of disjunctive inequalities (which is often the case in scheduling) [Tsang 95].

### 2. 3. 2 Bottleneck Analysis

The research in bottleneck analysis is motivated by the fact that focussing on the bottleneck resource first would have a greater impact on the quality of the solution. Two such approaches are Shifting Bottleneck Procedure [Adams 88] and OPT [Goldratt 80] [Fry 92]. The shifting bottleneck procedure involves solving a one-resource relaxation of the original

problem while continuously reoptimizing the schedule of the resources that were sequenced in earlier relaxations. The OPT system first identifies the bottleneck resource. Then it schedules the activities on this resource. After that, it schedules the resources upstream of bottleneck and then the resources downstream of the bottleneck. One of the drawback of OPT is that it can not deal with multiple or shifting bottleneck resources.

### 2. 3. 3 Dispatch Rules

Another approach to scheduling is to use priority dispatch rules. Whenever a machine becomes free, all schedulable activities are evaluated by a priority heuristic. The highest priority activity is then dispatched. Several priority heuristics are developed. [Panwalkar 77] provide a comprehensive survey of research dealing with scheduling rules, identification of the types of problems and measures of performance. [Blackstone 82] have critically reviewed the literature on dispatching rules. They mentioned the lack of good, uniform and broadly defined performance measures to evaluate the rules.

## 2. 4 Constraint Satisfaction Problem

In order to provide a context for discussion of the relevant literature in constraint-based scheduling we will first provide a brief overview of the constraint satisfaction problem.

A Constraint Satisfaction Problem (CSP) "is a problem composed of a finite set of variables, each of which is associated with a finite domain, and a set of constraints that restricts the values the variables can simultaneously take. The task is to assign a value to each variable satisfying all the constraints"[Tsang 93]. More formally, a CSP problem consists of:

- a finite set of variables $\{x_1, x_2, \ldots\ldots, x_n\}$ and their corresponding domains $\{D_{x1}, D_{x2}, \ldots\ldots, D_{xn}\}$. The domain is the set of all possible values that the variable can take.

- a finite set of constraints $\{c_1, c_2, \ldots\ldots, c_m\}$ defined on arbitrary set of variables which satisfies simultaneous assignment of values to the concerned set of variables from their respective domains.

A solution of the CSP refers to the assignment of values to all the variables from their respective domains that satisfies all constraints. The most common approach towards solving a CSP problem is to represent it as a graph where the variables are the nodes and the constraints are arcs. The solution process involves searching the graphs in a depth first manner. In the depth-

first search, a solution is incrementally built by assigning values to one variable at a time. Every time a variable is assigned a new search state is created that incorporates new constraints as a result of the assignment. If the problem reaches a state where it cannot assign values to any unassigned variables without violating any constraints or previous assignments, then the problem is said to reach the dead end. To recover from the deadend the problem backtracks to an earlier state and tries alternative variable assignment from its domain. The most common form of backtracking is called chronological backtracking, whereby the most recent assignment is undone and another alternatives are tried. The search stops when it finds the first solution, or when all alternatives from the domain of the variables are tried without success (the problem is infeasible) or any other pre-defined stopping criteria are met.

### 2. 4. 1 Improving the Search Efficiency of a CSP

The general CSP is NP-complete, which means that the search space of the problem grows exponentially with the problem size. Thus the backtrack search as described earlier may take exponential time in worst case. To improve the efficiency of the basic search process several techniques have been proposed. These techniques include consistency checking, variable and value ordering, and different variations of backtracking techniques such as backjumping and backmarking. For an overview of these techniques please refer to [Kumar 92][Beck 94][Sadeh 91]. We will briefly focus on two of these techniques which are related to our work.

### 2. 4. 2 Consistency Checking

The consistency checking refers to the propagation of the effects of a variable assignment to unassigned variables. The motivation behind this is to prune the search space by removing inconsistent assignments that cannot participate in future solution given the state of search [Mackworth 86]. This is achieved by inferring new constraints and propagating them to all unassigned variables whenever a variable is assigned a value. This propagation is based on three consistency algorithms. These are node consistency, arc consistency and path consistency [Mackworth 77]. The node consistency ensures that a variable's domain does not contain any value that does not satisfy unary constraints on that variable. The arc consistency ensures that two variables which are connected by binary constraint has only compatible values in their respective domain. Thus if two variables $x_a$ nd $x_b$ are connected by a binary constraints, then achieving a arc consistency would guarantee that any value in the domain of $x_a$

will have a counterpart in the domain of $x_b$ that satisfies the constraint between these two variables. Path consistency ensures that any number of variables that are mutually connected by constraints has no incompatible values in their respective domains.

### 2. 4. 3 Variable and Value Ordering

The order in which variables are instantiated and values are assigned have significant impact on the performance of the general backtrack search [Kumar 92]. A perfect variable and value ordering for a CSP problem that has a satisfying solution would produce a backtrack free search [Beck 94]. This motivated a significant number of researchers to investigate variable and value ordering heuristics [Haralick 80, Dechter 88, Dechter 89, Fox 89]. The most common approach is to find the most critical variable that is most difficult to instantiate and assign the least constraining value from its domain.

## 2. 5 Constraint-based Scheduling Methods

The application of constraint framework to scheduling problem was first demonstrated by [Fox 83] in ISIS scheduling system. ISIS used the knowledge of constraints to guide the search process. ISIS is the first scheduling system that attempts to deal with a wide array of constraints prevalent in a typical job-shop. The success of ISIS in solving real life problems has motivated a number of researchers to explore the applicability of AI techniques for scheduling problems. Given the wide variety of these research it is difficult to classify them into some general framework. For the sake of our discussion, we broadly classify them into the following categories:

### 2. 5. 1 Constructive Scheduling

Constructive scheduling refers to the approach of incrementally building solution by assigning consistent start times to one activity at a time. Several scheduling systems that have been reported in the literature that falls into this category. The first of such systems is ISIS which has already been mentioned. Despite the success of ISIS in real life application domain, it's job-centered approach fails to acknowledge the effect of bottleneck resources on scheduling. In this approach all activities in a job is scheduled first before moving into another job. This motivated the research into 'resource centered' approach.

The outcome of this is OPIS [Smith 89] scheduling system, which features a more flexible architecture in which the system reasons opportunistically as it constructs a plan. At each planning step, the system determines the most important decision to make from different perspectives, most notably order-based and resource-based. It combines the 'job-centered' approach of ISIS for non-bottleneck operations with 'resource-centered' approach for bottleneck resource. The ability to switch back between these two approaches gives rise to the idea of 'opportunism' in search, which has shown to be significantly improve the performance of the system as compared to ISIS. However, the opportunistic behavior of OPIS is tied to scheduling bottleneck *resource*. This coarse grained approach is termed as 'macro-opportunistic' approach as compared to a finer grained approach of 'micro-opportunistic' search [Sadeh 91]. The micro-opportunistic search incorporates an activity-centered search process whereby each activity is considered to be an independent decision point. The activity-based approach takes both jobs and resources and their interactions into account from a single perspective. The advantage of this approach lies in its ability to dynamically track down the emerging bottleneck in a more efficient manner than OPIS and shifting its attention towards this. The introduction of this micro-opportunistic approach has a significant impact on the solution of job-shop scheduling problems [Sadeh 91].

The research in constructive scheduling domain is mainly focussed on finding efficient variable and value ordering techniques [Sadeh 91] and alternative forms of backtracking techniques [Sadeh 94]. Among these approaches, our work is greatly influenced by the variable selection heuristic proposed by [Sadeh 91]. The heuristic, called the reliance-contention based heuristic tries to find the most critical resource based on the demand placed by various activities using the resource. Once the critical resource is identified it then finds the critical activity that places the maximum demand at a given time interval on the critical resource. The time interval is defined as the average duration of all the activities on the concerned resource. The advantage of this approach lies in its ability to dynamically identify the emerging bottleneck (critical resource) and focusing the search effort to reduce contention on this resource.

Another line of research within the constructive scheduling work is focussed on making sequencing decision instead of assigning start times [Bensana 88] [Muscettola 93] [Smith 93]. The objective is to post precedence constraints among activities contending for a critical resource thereby relieving the contention on it. This is a least-commitment approach as compared to start time assignment as it gives a family of solutions instead of just a single solution. This gives the flexibility of assigning alternative start times to activities at the execution

level thereby giving the opportunity to deal with unanticipated events. [Muscettola 93] and [Smith 93] have reported strong results compared to the [Sadeh 91]'s constructive approach on a set of benchmark problems. We are motivated by the success and inherent flexibility of the constraint posting approach as compared to the start time assignment approach. Our work combines the variable ordering heuristic of micro-opportunistic search with constraint-posting approach.

### 2. 5. 2 Repair-based Scheduling

In repair-based scheduling approaches, the search starts with a state where all variables are assigned and iteratively improves on the original inconsistent solution. GERRY [Zweben 94] and SPIKE [Johnston 89] are two such systems. SPIKE uses the Min-Conflicts heuristic proposed by [Minton 92] and was used in scheduling observations for the Hubble Space Telescope. The underlying philosophy of this approach is to repair the inconsistent values that reduces the conflict most. [Minton 92] has reported a strong results on N-queens problem. However, the quality of the solution of the repair-based approach is a function of the initial starting assignment. The philosophy behind the repair based approach is of interest to us. As would be shown later, in our approach to solve alternative scheduling problem, we start with an initial inconsistent assignment of 'probability of existence' values to each activities. During the course of the solution process we iteratively revise these values.

Although these two different approaches have shown strong results in their respective applications, no effort has been reported to date that combines these two approaches in solving a single type of problem. Due to this, two different research streams have been emerged among the constraint-based scheduling community whereby each group is focussed on improving the efficacy of their respective approach. Our view is that by combining the constructive approach with repair based approach in a single problem we will be able to expand the frontiers of constraint-based scheduling to new application domain. We will demonstrate in latter chapters how we combined these two paradigms to solve scheduling and resource management problems.

## 2. 6 Combining AI and OR

Another recent trend in scheduling literature combines the two paradigms of Artificial Intelligence (AI) and Operations Research (OR). The traditional focus of OR approach involves in achieving a high level of efficiency in its algorithms whereas the focus of AI research is to

investigate more general problem solving models and to solve the problems by using these models [Baptiste 95]. The research effort in combining the AI and OR algorithms tends to combine efficient algorithms for solving wide range of problems. An example of such approach is the incorporation of edge-finding algorithm with the constraint-directed search as described in [Nuijten 94] and the incorporation of branch-and-bound backtracking search with constraint propagation as reported in [Baptiste 1994]. The results of the experiments from [Nuijten 94] indicates a strong performance on a number of benchmark problems that were reported in the OR literature.

## 2. 7 Research in Inventory Management

Inventory management combines two distinct functions: inventory planning and inventory control. The purpose of any inventory management system is to provide answer to the following three basic questions [Baker 93]:

- What to order?
- How much to order and?
- When to order?

It's output drives both manufacturing and purchasing function in the enterprise. Inventory management in a manufacturing enterprise encompasses raw materials, components, finish goods and in-process inventory. The objective of inventory management is to ensure that neither the production nor the delivery of the finished goods to the customer is hampered by the shortage of any items (raw materials, components, sub assemblies or finished goods). The decision is constrained by the organization's objective to minimize inventory so as to reduce the cost associated with the holding of the inventories.

The market demand is the driving force behind all inventory management activities in an enterprise. The demand for a product is categorized as either dependent or independent. Independent demands are those which are uncertain and they needs to be forecasted. Dependent demands are generated either thorough the customer order (as in the case of make-to-order environment) or from the demand of their parent items. The demand for parent items are either forecasted or specified by customer orders. The selection of an appropriate inventory management technique is determined by the distinction of these two demand types.

The traditional approach for inventory management involves using Material Requirement Planning (MRP) systems for dependent and known demands, and order point based inven-

tory models for the independent, unknown or uncertain demands. Thus all the inventory management techniques that are discussed in the literature can be grouped into the following two classes:

- Order Point
- Material Requirement Planning

These models more or less answer the previously mentioned three fundamental questions of inventory management.

## 2. 7. 1 Order Point System

One of the earliest and most-researched inventory control system is the order point system. Order point, also called statistical inventory control and stock replenishment [Plossl 94], generates orders for parts whenever the stock falls below certain predetermined point. These models are only applicable for independent demands (as it ignores the existence of a part in an assembly) where the demands are forecasted based on previous demands. The most basic order point system is the EOQ (Economic Order Quantity) model which determines the optimum order quantity that minimizes the cost of ordering and inventory holding costs. A considerable amount of effort is directed by the academicians to develop different variations of order point model. An excellent summary of the evolution of inventory research and the highly diversified research trends of this field can be found in the publication of [Aggarwal 74] and [Chikan 90].

A review of all these models revealed the following drawbacks:

- The model's assumption that the demand can be forecasted with reasonable accuracy is not quite true. The uncertainty of the market demands often invalidates this assumption.
- One of the major drawbacks of the 'Order Point' based models is their inability to deal with "dependent demands". These models ignore the interdependencies between the components parts. This can be explained by a simple example. Let us consider that we have a finished goods that requires three component parts. Let our OP model guarantees that, independently all these components would be available 95% of the time (because of the stochastic nature of the model we have to provide higher safety stock to guarantee this 95% availability). Now the probability that all these three components would be available at any time (for the assembly of their parent component) is 0.95 x 0.95 x 0.95 = 0.85. That is, even with a 95%

probability for the availability of individual components we can say that all these components would be available simultaneously only 85% of the time. Thus 15% of the time we would not be able to assemble the parent component due to unavailability of one of the three components parts.

Such uncertainties often deteriorate the effectiveness of the entire manufacturing plan. This prompted the search for new models that can guarantee the availability of the component items at the time of assembly. Materials Requirement Planning (MRP) is the outcome of such need. It was first successfully applied by J.A. Orlicky in 1961 on J.I.Case Company farm machinery [Plossl 94].

## 2. 7. 2 Materials Requirement Planning (MRP)

MRP was first introduced by Orlicky in 1975. The premise under which MRP was introduced was to ensure the availability of "the right parts, at the right time, in the right quantitites"[Wright 74]. MRP system is viewed both as materials planning and scheduling tool. For materials planning, it uses bill-of-materials explosion to generate the demands for each components corresponding to a finished product/sub-assembly. It's objective is to answer the following questions:

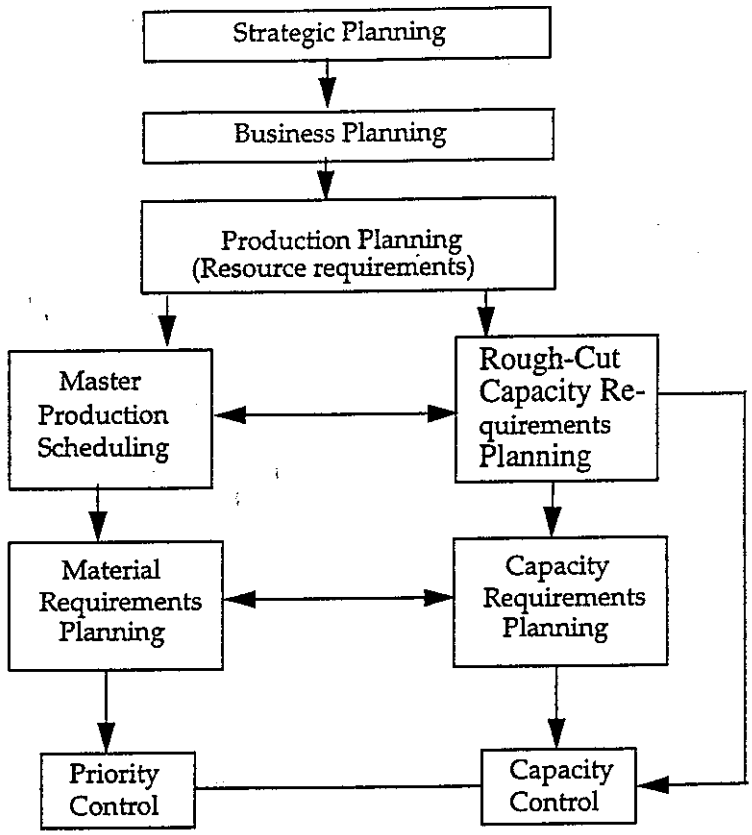
- How much to make?
- When to make?
- How much to buy?
- When to buy?
- How much to draw from inventory?

Ever since its introduction to the industry MRP draws a widespread attention. This widespread popularity of MRP can be attributed primarily to the simplicity of its logic and the need for a tool to overcome the problem of production interruptions due to component shortages. The output of the MRP is a set of time phased manufacture/assemble and purchase orders. The scheduling aspect of MRP refers to the assignments of time points to these orders. The time points specified in these orders act as a bound to the subsequent scheduling processes.

The intent of MRP is to manage resources within the enterprise effectively. In order to understand what it means "to manage effectively" we need to provide a basis as to where the MRP functions fits into the operations of a manufacturing organization. As shown in the figure 2.1, MRP is driven by the output of the Master Production Schedule (MPS). The MPS decides on

a periodic (monthly or weekly) basis of how much to produce of each of the finished items. Once the finished goods production quantities and the time when they are to be produced are determined the next step is to generate Material Requirement Plan (MRP) corresponding to each of the finished components.

**Figure 2.1. Stages of production planning**




MRP uses bill-of-material explosion to generate the demands of each components corresponding to a finished product. The first level of explosion determines all the subassemblies that are needed. The subassemblies are then broken into second, third, and so on levels of subassemblies until, at the lowest level, only purchased items exist. MRP considers the inventory at hand, scheduled receipts and calculate the net requirements that either needs to be purchased or manufactured. The output of the MRP is a set of production order and purchase orders.

Purchase orders are processed to procure items from the suppliers and the manufacturing/assembling orders are sent to the shop floor. The manufacturing/assembling orders then

drive the scheduling process at the shop-floor level. Scheduling involves allocating resources to tasks or activities over time. Schedulers use the manufacturing orders, to generates process plan and schedule the activities in the process plan in order to meet the due dates specified in the manufacturing order. Various techniques are used by the schedulers to schedule the order. Depending on the complexity of the shop-floor this might vary from manual scheduling based on scheduler's intuition and pervious experience to sophisticated dispatching rules or simulation techniques.

## 2. 7. 3 Shortcomings of the MRP Systems

Despite its widespread adoption, few implementations are successful and MRP literature concedes that there are few "Class A" users, i.e. the users who follow every decisions given by MRP [Baudin 90]. There are several shortcomings of the MRP systems that can be attributed to the ineffectiveness of the MRP systems. The inherent dynamics of the supply chain invalidates many of the assumptions that are embedded in the MRP. These assumptions not only undermine the solution quality but are also unrealistic in many circumstances. The sequential nature of the MRP planning overlooks many of the downstream constraints at the planning stage. The following sections identify these shortcomings:

### 2. 7. 3. 1 Infinite Capacity Assumption

MRP systems generates manufacturing and purchase orders on the basis of infinite capacity. Initially, it was justified on the premise that, MRP is a planning tool and capacity consideration at the planning level would overwhelm the entire process. Thus MRP module left the capacity considerations for subsequent execution levels. This assumption imposes enormous burden at the execution levels. For example, it might appear at the shop-floor level that the order generated by the MRP systems can not be completed within the specified due date due to resource capacity constraints. But, it becomes too late to modify the due date as other subsequent processes are generated on the basis of the initial order. Thus, if a due date for a subassembly can not be met because of schedule infeasibility, this might result in the build-up of work in-process (WIP) inventory for other sub-assemblies waiting for that particular product. The late production of a part may also invalidate the entire production plan for its parent components resulting in missed due dates. Thus the failure to take into account the execution level constraints at the planning stage has a propagating effect in case of execution level failure and this might interfere with the planning processes at other level.

In order to over come this problem, MRP was evolved as MRPII (Manufacturing Resource Planning), whereby consideration of resource capacity is done at the planning level. But even for MRPII, the capacity constraints are considered at a very abstract level. It checks the capacity violations based on the average demands placed on different machines. The gross generalization of such rough-cut capacity planning ignores the effect of detailed scheduling on the resource capacity. Thus, even with the introduction of capacity planning module, MRPII systems failed to take into account the requirements of capacity in a realistic manner.

### 2. 7. 3. 2 Fixed Lead Time

Another shortcoming of the MRP system that invalidates its execution is its assumption about activity lead times. It assumes fixed lead times both for manufacturing and purchasing activities and it generates the manufacturing and purchasing orders on the basis of these lead times. In reality, an organization might have some degree of control over the manufacturing lead time, it has a much lesser degree of control over purchasing lead time. Purchasing being an activity associated with entities outside of an enterprise, is more prone to lead time fluctuations. Thus, the variation of lead times is a norm rather than an exception, and the assumption of a fixed lead time at the planning level might invalidate the plan at the execution level.

### 2. 7. 3. 3 Poor Inventory Management

One of the requirements of any resource management system is that, it should be able to plan and manage inventory effectively. Looking back to MRP systems, we find that it fails in this account heavily. As already mentioned, MRP uses fixed lead times for planning orders. But these lead times vary due to the inherent uncertainty associated with the manufacturing and purchasing activities. To hedge against this uncertainty in lead times, MRP system allocates safety stocks. The safety stock is calculated on the basis of the variability of the lead times. Thus more variable a lead time is, the more safety stock is allocated to the associated products or components. This safety stock places an additional burden on the inventory that an enterprise carries.

### 2. 7. 3. 4 Inflexibility in Decision Making

The fourth shortcomings of the MRP system is its inherent inflexibility in decision making process. A good planning and scheduling tool must consider all available alternatives before making its decision. But MRP ignores this aspect of planning. As mentioned earlier, the sources of these alternatives are varied. There could be alternate machines, alternate raw

materials, alternate process plans, alternate suppliers, alternate operators etc. From a resource planning/management perspective, alternatives mainly originates from different ways of satisfying a demand. The MRP logic does not take into account such varied sources of alternatives. This unnecessarily constrain the decision making process. For example, if a MRP system detects through its rough-cut capacity planning that a manufacturing order is not feasible for the given due date, it does not consider other alternatives, e.g. purchasing from an outside supplier or sub-contracting to an outside manufacturer. This results in poor quality decisions.

Despite the above shortcomings, MRP systems are still considered to be the most popular resource management tool in the discrete part manufacturing environment. Part of this is attributed to its simplicity, and part of it is due to the unavailability of any alternative tools.

## 2.8 Conclusion

This chapter provides an overview of the approaches to resource management. We view resource management as the function that encompasses activities pertaining to scheduling and resource management. The review covers both traditional and recent approaches towards the problem scheduling and inventory management. The major shortcoming of the current approaches is that they fail to acknowledge the interaction between these two problems. Another shortcoming of the existing modelling techniques are their inability to capture wide variety of constraints associated with these two problem domains. Also, the lack of generality of the existing models forces the modeler to look for domain specific models. As each of these domain specific models is built on its own domain dependent assumptions, they often fail to give the expected results when put into operations with other domain specific models. Examples can be drawn from scheduling and MRP systems. Whereas in scheduling the constraints on resource capacity is recognized, MRP tend to ignore this constraint. The results of this is conflicts at the execution level. This calls for a more robust modeling framework that will allow the modelling of the resource management problems in a wide variety of domains. A modeling formalism that is uniform across organizational tiers and which is applicable at various levels of decision making is definitely going to improve the quality of decision making related to resource management.

# *Chapter 3 Modeling Alternative Scheduling Problem*

This chapter describes our approach towards modeling the alternative scheduling problem as a constraint satisfaction problem (CSP). Our motivation to deal with such problems came from the fact that, the resource management problem can be modeled as an alternative scheduling problem, and modelling the alternative scheduling problem as a CSP problem is a leap towards solving the resource management problem. We will deal with the issue of solving the alternative scheduling problem in the next chapter and we will then extend the problem solving framework to solve resource management problems in subsequent the chapters of the report. In the first section of this chapter, we define the "alternative" scheduling problem. Modeling issues are discussed in the subsequent sections.

## 3. 1 The Alternative Scheduling Problem

Job-shop scheduling involves scheduling set of jobs on set of machines (resources). The objective is to assign release and due dates to all jobs by satisfying the associated constraints. Scheduling problems of this class have proven to be NP-complete [Garey 79]. The complexity of the scheduling problems is attributed to the combinatorial search space associated with these problems. Eighty-five orders moving through ten operations without alternatives, with a single substitutable machine for each operation, and no machine idle time, has over $10^{1000}$ possible schedules [Fox 94]. The existence of alternative choices in such problems further complicates the solution process, as it increases the search space further.

A few attempts to model and solve scheduling problems with alternative choices have been reported in the literature [Miyashita 94][Nuijten 94]. Most of these are focussed on solving scheduling problems with alternative resources. Very little research effort was directed in the past to solve a broader class of scheduling problems that involve alternative activities or alternative process plans. Our effort in this line was motivated by these shortcomings. We are exploring methods that are generic enough to represent and solve different classes of alternative scheduling problems. More specifically, we are interested in solving the following four different classes of problems. The list of notations shown in figure 3.1 will be used for describing scheduling problems in the subsequent sections and the chapters of this report.

Note that, the terms 'duration' and 'processing times' are used interchangeably in the subsequent discussions.

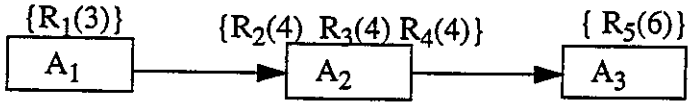
**Figure 3.1. List of notations**

$A_i$ = Activity i
$d_{ij}$ = Processing time(duration) of Activity i on Resource j
$R_j(d_{ij})$ = Resource j with processing time $d_{ij}$
EST = Earliest Start Time
LST = Latest Start Time
EET = Earliest End Time
LET = Latest End Time

## 3. 1. 1 Alternative Resources with Same Duration

This type of problem specifies that, an activity can be executed by any one of the available alternative resources. The processing times for these resources are same. Thus, the alternative resources can be considered to be identical. The activity network of figure 3.2 specifies a simple instance of such problem:

**Figure 3.2. Alternative resources with same duration**

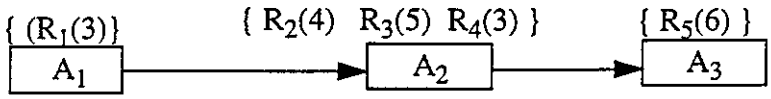



The network specifies that we have three activities $A_1$, $A_2$ and $A_3$. Activity $A_1$ uses resource $R_1$ has a processing time (duration) of 3. Activity $A_2$ has a processing time of 4 and can use either resource $R_2$, $R_3$ or $R_4$. The processing time of $A_2$ remains the same regardless of the resource used. Thus for the activity $A_2$, the three resources are identical. Activity $A_3$ has a processing time of 6 and uses resource $R_5$.

## 3. 1. 2 Alternative Resources with Variable Duration

This type of problem specifies that an activity can be executed by any one of the available alternative resources (e.g.machines), but the processing time on these resources are different i.e., the resources are non-identical. An example of such problem is encountered when we have a technologically advanced machine, which requires less processing time compared to its less-sophisticated counterpart. The activity network of figure 3.3 specifies a simple instance of such problem:

Figure 3.3. Alternative resources with variable duration

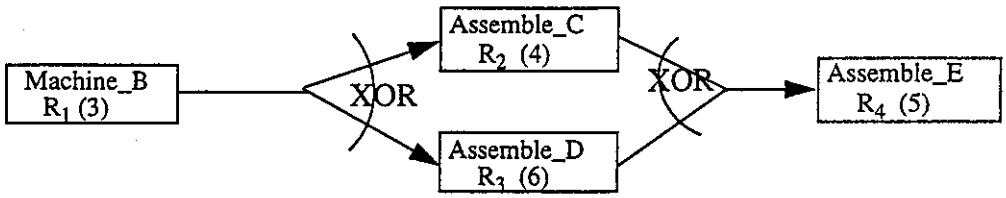



The network specifies that, we have three activities $A_1$, $A_2$ and $A_3$. Activity $A_1$ uses resource $R_1$ and its duration is 3. Activity $A_2$ can use either resource $R_2$, $R_3$ or $R_4$ with processing time of 4, 5 and 3 respectively. Activity $A_3$ uses resource $R_5$ and has a duration of 6.

## 3. 1. 3 Alternative Activities

The third class of problems that we are interested in, is scheduling problems with alternative activities. Alternative activities can exist as a result of using alternative machines with variable duration (as described in the preceding section) or it can represent a completely different type of problem. For example, an alternative to manufacturing could be sub-contracting or purchasing, or a part could have an alternative components in its bill-of-material. This type of problem can also be represented as an alternative activity problem. An instance of such problem is represented in the figure 3.4, which specifies that before we execute the activity

Figure 3.4. Alternative activities




Assemble_E, we have to first execute activity Machine_B and then either of the activities Assemble_C or Assemble_D. Component C and D are the alternative components of part E. The activity Machine_B uses resource $R_1$ and its duration is 3. The activity Assemble_C and Assemble_D uses resource $R_2$ and $R_4$ respectively and their duration is 4 and 6 respectively. Activity Assemble_E has a duration 5 and uses resource $R_4$.

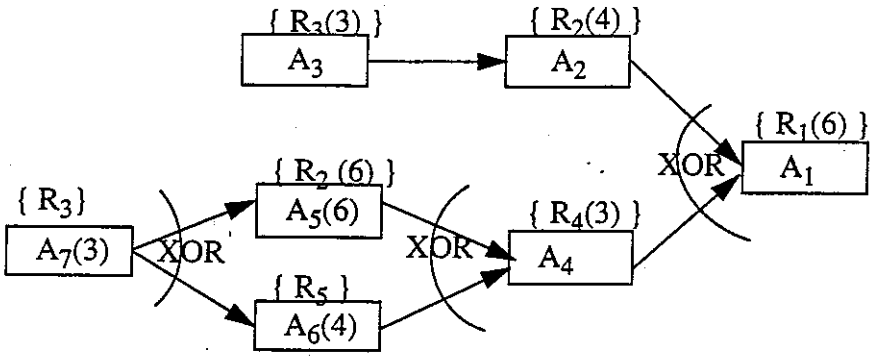
## 3. 1. 4 Alternative Process Plan

Alternative process plan problems specify that a job can be completed by executing different sequence of operations. There is a distinction between the alternative activity and alternative process plan problems. Whereas an alternate activity specifies a disjunction involving a sin-

gle activity, alternative process plan can specify disjunction involving a *set* of activities. For example, the problem shown in figure 3.5 specifies that in order to execute activity $A_1$, we have to either:

- execute activity $A_3$ and then activity $A_2$ OR,
- execute activity $A_7$, then either activity $A_5$ or $A_6$, and then activity $A_4$.

**Figure 3.5. Alternative process plan**




Note that, we could have alternative activities in the alternative branches of the process plan as is the case for $A_5$ and $A_6$ in the above network.

It is this class of alternative scheduling problems that can be used for resource management decisions. The mapping of such problems gives rise to two major issues. First of them is the representational issue. We need a representation framework that is generic enough across a wide range of problems and which in conformance with the constraint based problem solving framework. The another issue is, how to solve these problems. As the effectiveness of resource management is directly dependent on the quality of decision making concerning these alternative choices, we need some ways to discriminate among these choices. Can the existing problem solving techniques in constraint based scheduling paradigm be used to model and solve these problems, or do we need to develop new methods for solving them? These issues are addressed in the subsequent chapters of the report.

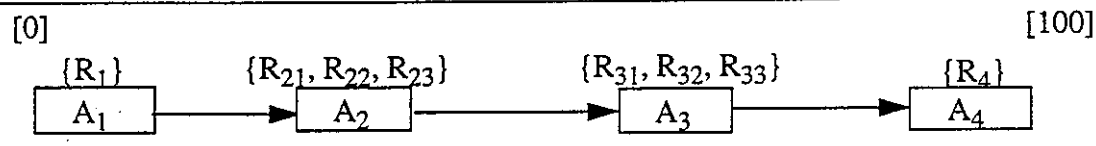
## 3. 2 Modeling the Problem

Since the current research is based upon the premise that CSP framework can be effectively used to address the resource management problem, we want to model our problem as a constraint network. The mapping of alternative decision variables gives rise to the disjunctive

nodes in the network. The problem of disjunctiveness has some of the subtle intricacies that are not generally addressed in the CSP literature. Thus we are lacking a thorough understanding of the complexities associated with such problems. In order to get an understanding of the complexities associated with such problems, we will start with a simple job-shop scheduling problem with alternative resources. The disjunctiveness in the problem arises from the fact that, a single activity can use any resource from a set of alternatives. We plan to model the problem as *a set of alternative activities*, each of which uses one of the resources from the alternative resource set. We are assuming that the disjunctions are of type XOR (Exclusive OR).

The problem, as shown in figure 3.6 is a single job scheduling problem. The activity network shown in the figure corresponds to the process routing of this job. The network specifies that, we have four activities $A_1$, $A_2$, $A_3$ and $A_4$. Activity $A_1$ requires(uses) resource $R_1$, activity $A_4$ requires $R_4$, and activity $A_2$ and $A_3$ each have three alternative resources to choose from. The activities ($A_2$ and $A_3$) will use any one (exclusive OR) of the resources from the set of its alternatives. There are temporal constraints between these activities. $A_1$ must precede $A_2$, $A_2$ must precede $A_3$, and $A_3$ must precede $A_4$. The values in the square brackets [0] and [100], is the earliest start time and the latest finish time (due date) for the order.

**Figure 3.6. : A single job scheduling problem with alternative resources**

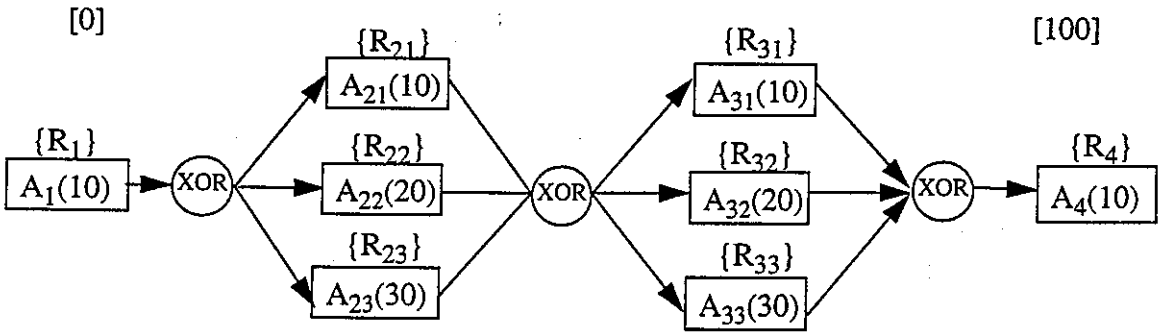



The disjunctiveness in the above network arises from the fact that activities $A_2$ and $A_3$ can use any one of the three alternative resources specified in their respective resource alternatives. If we assume that the duration of the activity $A_2$ remains the same for all of its alternative resources, then the temporal propagation through the activity approaches more or less to that of the temporal constraint propagation algorithm of ODO [Davis 94]. However, if the duration of the disjunctive activities varies, we can no longer use the existing temporal propagation algorithm. This is due to the fact that, while propagating through the activities $A_2$ and $A_3$ we would have to use the duration of the respective activities, and since we have three different durations for each of these activities, we don't know which duration to use. If we use the smallest duration (the resource with the smallest processing time), then we will

rule out the possibility of using other resources at the first place, as this would trim off some of the possible end times of the other alternatives. Whereas, if we use the largest duration, we will unnecessarily over-constrain the end-times of other alternatives.

In order to deal with this problem, we model the disjunction as a set of three activities requiring three different resources. Our motivation behind such representation is, we can use the similar representation for other classes of alternative scheduling problems involving alternative activities or alternative process plans. And this would allow us to use the same problem solving methodology to solve scheduling problems with alternative resources as well as alternative activities. Moreover, this representation is also in conformance with the micro-opportunistic search approach (whereby each activity is considered to be a decision point), which has reported to provide strong performance performance results for job shop scheduling problems [Sadeh 91]. With this representation, our initial problem now can be represented in the following form (figure 3.7). The values within the parentheses indicates the respective duration of the activities.

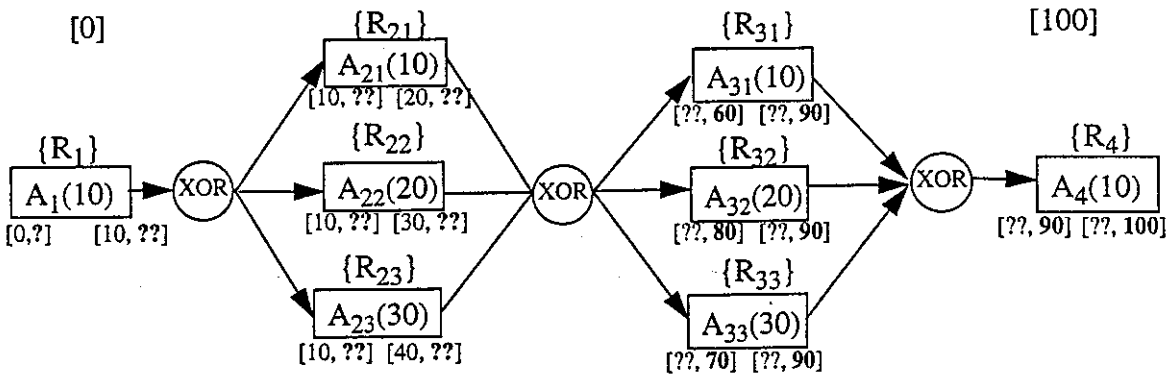
**Figure 3.7. Single job scheduling problem with alternative activities**




## 3. 3 Problems with Temporal Propagation

Given the above representation, our next task is to address the issues related to temporal propagation through the disjunctive activity network. For the ease of discussion, we introduce two notations. Let $\{A_2\}$ represents the set of activities $A_{21}$, $A_{22}$ and $A_{23}$. Similarly $\{A_3\}$ represents the set of activities $A_{31}$, $A_{32}$ and $A_{33}$. We will use the same notations in the subsequent sections of this chapter. For temporal propagation, we will use the forward and backward temporal propagation algorithm as described in [Davis 94].

Figure 3.8. **Temporal propagation in a single job scheduling problem with alternative activities**




The temporal propagation starts from the activity $A_1$. First we propagate the minimum values of the start time interval. Starting from activity $A_1$, we propagate the values forward along the network. While propagating the values across $\{A_3\}$ we encounter a problem. Which Earliest End Time (EET) value from $\{A_2\}$ are we going to assign to the start time interval of $\{A_3\}$? Would it be 20, 30 or 40? Similarly, while propagating backwards from $A_4$, which Latest Start Time (LST) value of $\{A_3\}$ are we going to assign to the Latest End Time (LET) of activities $\{A_2\}$? Would it be 80, 70 or 60. How would our propagation mechanism discriminate among these values? If we use binary temporal constraints among the activities and propagate the values across it, we would not be able to make such decisions. We need to have a mechanism that would allow us to decide which value to propagate in situations like this. For this, we propose the addition of a new type of constraint to our activity network.

## 3. 4 N-ary Constraint (NaC)

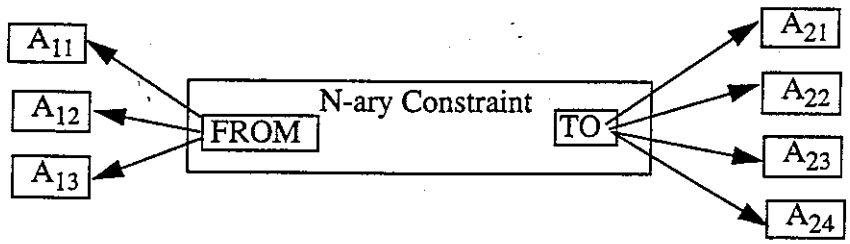
Binary temporal constraints links *two* temporal variables (e.g. start time variable of one activity to the end time variable of another activity). But for our purpose, we need to have a constraint that is capable of linking $m$ number of temporal variables to $n$ number of temporal variables (where both $m$ and $n >= 1$). For this, we propose to use N-ary constraints (NaC). The NaC binds the temporal variables of all the activities in a disjunctive set (like the one we discussed earlier) to the activities 'before' or 'after' the disjunction. Thus, whenever we need to establish temporal constraints with an activity having alternatives (either resource alternative or activity alternative), we will incorporate a NaC between them.

In addition to the capability of handling the propagation along the disjunctive activities, the introduction of NaCs generalizes our representation of disjunctive nodes. The activity nodes can be used, as usual, as a decision point for any heuristics that we might use in our problem solving.

## 3. 4. 1 Design of the N-ary Constraint

The proposed constraint is comprised of the following structure. As shown in the figure 3.9, the constraint maintains two pointers to the lists of activities that are constrained by the constraint. The 'From' points to the list activities which temporally precedes the list of activities

**Figure 3.9. Structure of the N-ary Constraint**

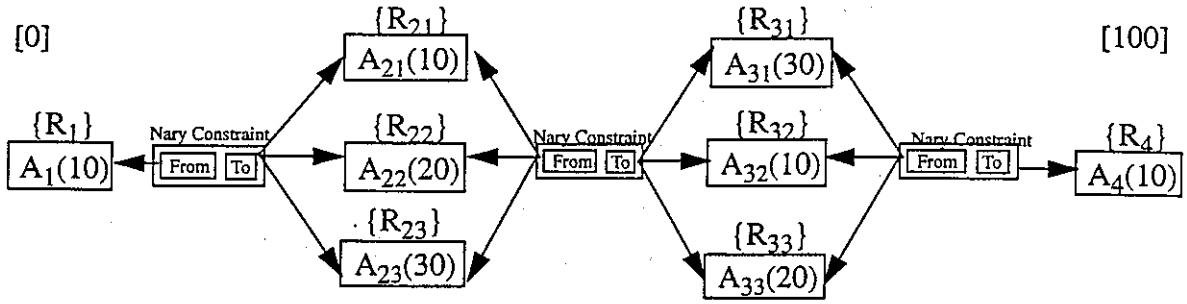



pointed to by the 'To'. In addition to these, there exists an interval variable (an interval variable is one which represents the domain of possible values for the variable) within the constraint that establishes the offsets between the two sets of activities pointed to by the 'From' and 'To'. The reason for providing such an offset interval variable is that, we want to express variable offset values between different activities. For example, we might want to model a problem which specifies that activity A starts "no sooner than the time unit 5 but no later than the time unit 6" after the activity B ends. An instance of such problem is described in [Cheng 95] where in a semiconductor manufacturing process the wafer has to be cooled certain time before it can be immersed in the chemical bath. But if it is cooled for more than certain specified amount of time, then the product is wasted. By expressing "offset" to be an interval variable we would be able to model problems like this. This type of problem instances are also in abundance in the process industries.

## 3. 4. 2 Propagation Algorithm For the Nary Constraint

With the addition of the new constraint, our network assumes the following form shown in the figure 3.10. Let's now discuss the propagation algorithm through the N-ary constraints. We will use the network of figure 3.10 for our subsequent discussion. Let's consider that we have a NaC of type 'before/after' between the activity sets $\{A_2\}$ and $\{A_3\}$ with offset zero.

**Figure 3.10. An activity network with N-ary temporal constraints (before propagation)**



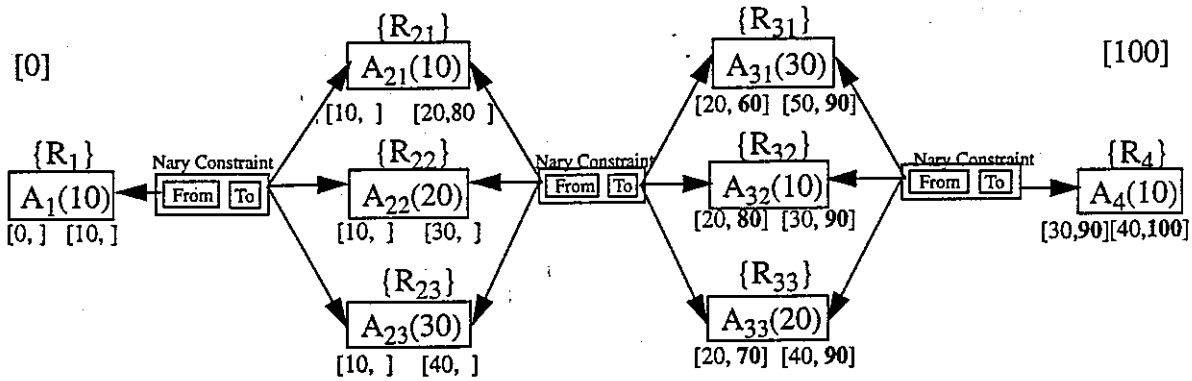

What this constraint specifies is that, whatever activity we choose from the set $\{A_3\}$, it should start after the activity that we decide to execute from the set $\{A_2\}$. Note that, this constraint is not equivalent to having a set of binary 'before/after' constraints between the sets of activities. Because, a binary constraint among these activities would ensure that ESTs of *all* the activities in $\{A_3\}$ is greater than or equal to the LETs of the *all* the activities of the set $\{A_2\}$. As we will show later, this would unnecessarily overconstrain the problem, and may lead to an erroneous decision.

In order to achieve arc-consistency in the constraint graph, we propagate the temporal values across our network. First, we propagate the earliest start (EST) and earliest end time(EET) during our forward propagation. We will explain the propagation algorithm by an example. Let's start with the activity $A_1$. We set the EST of $A_1$ to '0' and then propagate the value forward setting EET of $A_1$ to be equal to 10. Now we have to propagate the value through the NaC. Since the propagation comes from a set of activity comprising only one member ($A_1$) the propagation is straight forward. We set the EST of $\{A_2\}$ to be equal to 10. Propagating forward through these activities, we set the EET of $A_{21}$, $A_{22}$, $A_{23}$ to be equal to 20,30 and 40 respectively. Now, we have to propagate through the second NaC in the network. At this point, we have to decide which value to propagate? We have three EET values corresponding to the three activities $A_{21}$, $A_{22}$ and $A_{23}$.

These are 20, 30 and 40. As we do not intend to unnecessarily over-constrain the domain of any interval variable (thereby eliminating possible start or end times of an activity), we would propagate *MINIMUM* of all the EETs of $\{A_2\}$ which in fact contains the other possible intervals. Thus we set the ESTs of $\{A_3\}$ to be equal to min{20,30,40} = 20. By propagating the minimum values we ensure that the other possible intervals of the disjunction $\{A_2\}$ are con-

tained in the propagated value. Had we chosen the maximum value of 40, we would have eliminated the possible start times between 20 and 39. As at this point, we are not certain which alternative from the set $\{A_2\}$ are we going to choose, this would have unnecessarily constrained the start time domain of activities in $\{A_3\}$. Please note that, had we used binary temporal constraints between the set $\{A_2\}$ and $\{A_3\}$ it would have set the ESTs of $\{A_3\}$ to be equal to 40 whenever propagation follows from activity $A_{23}$. But our propagation algorithm ensures that no matter from which activity propagation comes, it will always propagate the minimum value of the EET from the disjunction.

**Figure 3.11 An activity network with N-ary temporal constraints (after propagation)**




Moving along, we set the EET of $A_{31}$, $A_{32}$, $A_{33}$ to be equal to 50, 30 and 40 respectively. Following the same logic, we set the EST of $A_4$ to be equal to 30 after propagating forward through the third NaC in the activity network.

While propagating backwards, we set the value of the Latest End Time (LET) of the activities $\{A_3\}$ to be equal to the LST of $A_4$ as the propagation is following from a set containing only one activity. But while propagating back from the LETs of $\{A_3\}$ we have to decide which value to propagate. Following the same logic (that we don't want to unnecessarily constrain the interval values) we propagate the *MAXIMUM* of all the LSTs of $\{A_3\}$ to the LETs of $\{A_2\}$. This sets the LET of $\{A_2\}$ to be equal to 80. We then continue our propagation backwards in the usual manner.

In the preceding discussion, we noticed that while propagating values through NaCs, depending on the direction of our propagation we need to decide which value to propagate. Thus while propagating temporal values forward, we need to propagate the minimum of the

end time intervals of the activities and while propagating temporal values backward, we need to propagate the maximum of the start time intervals of all the activities associate with the disjunction. This observation has prompted us to have two different propagation algorithm for forward and backward propagation. In addition to the notations introduced in figure 3.1, we will use the following notations for describing these algorithms.

```
Act_to = List of activities to which the propagation is following
Act_from = List of activities from which the propagation is coming
Act_o = Activity from which the propagation comes;
EET_o = EET of the activity from which propagation comes
LST_o = LST of the activity from which propagation comes
```

### 3. 4. 2. 1 Full temporal propagation

The full temporal propagation propagates values through the Nary temporal constraint. It guarantees full temporal arc-B-consistency regardless of the state of the consistency graph before the propagation.

**Forward Propagation:**

```
    /// first decide from which alternative activity to
    /// propagate the EET value

    if the number of activities in Act_from > 1
    for all activities in Act_from
    {
        if EET of the activity < EET_o
        EET_o = EET;
    }

/// now propagate the value

for all activities in Act_to
{
    set EST = EET_o;
    propagate the value internally to the EndTime variable.
    propagate the value of EET to downstream activities;
}
```

**Backward Propagation:**

```
    /// first decide from which alternate activity to
    /// propagate the LST value
```

```
   if the number of activities in Act_to > 1
   for all activities in Act_to
   {
      if LST > LST_o
      LST_o = LST;
   }

/// now propagate the value

for all activities in Act_from
{
   set LET = LST_o;
   propagate the value internally to the StartTime variable.
   propagate the value to upstream activities;
}
```

### 3. 4. 2. 2 Temporal propagation

If full temporal arc-B-consistency was held before the assertion of a new commitment that broke this arc consistency, the *temporal propagation* guarantees full temporal arc-B-consistency after this propagation has been performed. Notice that, this algorithm is slightly different than that of full propagation algorithm. For this case, during backward propagation, we first determine the activity having the highest LST (lowest EET for forward propagation). If the activity from which the propagation comes is different from the activity having the highest LST (or lowest EET for forward propagation), then we don't propagate any value as it won't have any effect for the upstream (or downstream for forward propagation) activities. But, in case of full temporal propagation, we do not perform checks similar to this before propagating the values.

**Forward Propagation:**

```
/// first decide from which alternate activity to propagate
/// the EET value

   if the number of activities in Act_from > 1
       for all activities in Act_from
       {
          if EET < EET_o
          EET_o = EET;
       }

       if EET_o is not equal to the EET of Act_o
          RETURN;

 // now propagate the value

   for all activities in Act_to
   {
```

```
        set EST = EET_o;
        propagate the value internally to the EndTime variable.
        propagate the value to downstream activities;
}
```

**Backward Propagation:**

```
if the number of activities in Act_to > 1
   for all activities in Act_to
   {
      if LST > LST_o
      LST_o = LST;
   }
   if LST_o is not equal to the LST of Act_o
      RETURN;

   for all activities in Act_from
   {
     set LET = LST_o;
      propagate the value internally to the StartTime variable.
     propagate the value to upstream activities;
   }
```

## 3. 5 Probability of Existence (PEX)

Besides the complexities associated with temporal propagation, the existence of alternatives in the constraint graph gives rise to other issues as well. For example, unlike the standard job-shop scheduling problem, not all of the activities of the initial activity network will exist in the final solution. As the type of disjunction that we are dealing with are of type XOR (Exclusive OR), only one of the alternatives will exist in the final solution. In the course of our search for the solution, we have to decide which activity will remain in the final solution and which will not (or which activity are we going to execute and which are not). We thus introduce the notion of existence of an activity in order to facilitate such decision. The new variable, called Probability of Existence (PEX) would be associated with each activity and is very similar to a status variable. The value of the variable, which lies between 0 to 1.0, indicates the state of existence of the activity in the network; 0 being the state indicating that the activity will not exist in the final solution and 1.0 being the state of certain existence in the network.

For scheduling problems, each activity is assigned a PEX variable. During the initial instantiation of an activity network, all activities in the activity network are instantiated with a PEX variable having a domain of (0, 1.0). When a domain of a PEX variable of an activity is modified as a result of a heuristic decision, the effect of such modification is propagated across the

network. The constraint through which such propagation takes place is discussed in the following section.

## 3. 6 Representing Existence Dependency

The existence of alternative activities in the constraint graph has another implication. Not all of these alternatives are going to exist in the final solution. This means that we will be pruning alternative choices. This gives rise to the questions like - does this pruning have any impact on other activities in the network? How does the pruning of one activity interfere with other activities in the activity network. In order to explain the scenario, let's consider that we have an activity network like the one shown in figure 3.12, which represents a process plan for certain manufacturing operations:

**Figure 3.12. Activity Network representing existence dependency**

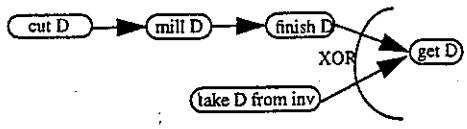



Figure 3.12 shows that, there are five activities in the network which are temporally constrained. The temporal constraint specifies that activity 'cut D' must be executed before the activity 'mill D' and the activity 'mill D' must be executed before the activity 'finish D'. We cannot start executing activity 'finish D' unless we have finished activity 'mill D' which in turn cannot be started unless we have finished activity 'cut D'. Thus the activity 'mill D' is *temporarily dependent* on activity 'cut D' and the activity 'finish D' is *temporarily dependent* on the activity 'mill D'. This is an example of temporal dependency. This sort of dependency is well captured by the temporal constraints between the activities.

Let's look at same network from another perspective. Note that, there exists a disjunct in the network. The effect of this XOR disjunct is that, either we will decide to execute the activity 'finish D' or we will execute the activity 'take D from inventory'. Now, if we decide to execute activity 'finish D', then we also have to execute activities 'mill D' and 'cut D'. On the other hand, if we decide to execute the activity 'take from inventory' then we will not be executing activity 'finish D'. And if we don't execute 'finish D', then we don't need to execute activities 'cut D' and 'mill D'. What this example shows is, the existence of activities 'cut D' and 'mill D' *depends* on the existence of activity 'finish D'. While activity 'finish D' is temporarily dependent on activity 'mill D', the existence of activity 'mill D' depends on the existence of

the activity 'finish D'. This shows that, the existence dependency and the temporal dependency is fully reversed for activities 'cut D', 'mill D' and 'finish D'. Thus, we cannot use the temporal constraint to express the existence dependency between activities. We need a different representation for this. Toward this end, we are proposing a new kind of constraint, called Existence Dependency Constraint (EDConstraint).

The following sections provide detail description of the design and the propagation algorithm for this constraint. Like the temporal constraints, we will need three different types of EDConstraints, namely unary, binary and n-ary. The following three sections treats these three types of constraints at a greater detail.

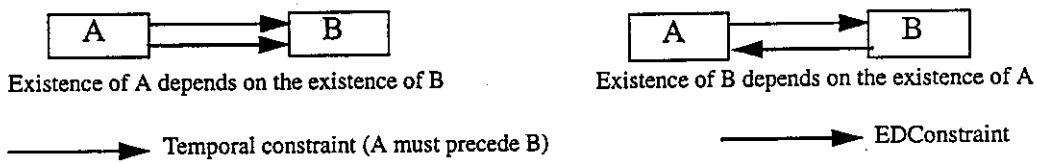
## 3. 6. 1 Unary EDConstraint

The unary EDConstraint enforces certain restrictions on the value of the PEX. For example, while instantiating the activity network we are certain about the PEX of certain activities. So we can set their values to be equal to the known value. Also, during the course of our problem solving we might assign new values of PEX to the activities.

## 3. 6. 2 Binary EDConstraint

Binary EDConstraint establishes dependency relationship between two activities. We can think of two possible cases for binary EDConstraint which are shown in the following figure:

**Figure 3.13. Binary Existence Dependency Constraint**




Let's consider the first case shown in the figure 3.13. There exists a temporal constraint between activities A and B which states that A must precede B. The EDConstraint establishes relationship that existence of activity A depends on the existence of activity B. If the existence A depends on the existence of B then we want to ensure that:

```
activity A exists when the activity B exists ----  (1)
and, activity B exists when the activity A exists ----  (2)
```

Logically the statement 1 is equivalent to:

```
B -> A.
```

Statement 2 can be asserted by the following logical implication:

```
A -> B.
```

Combining the above two implications, we can express the existence dependencies between two activities as:

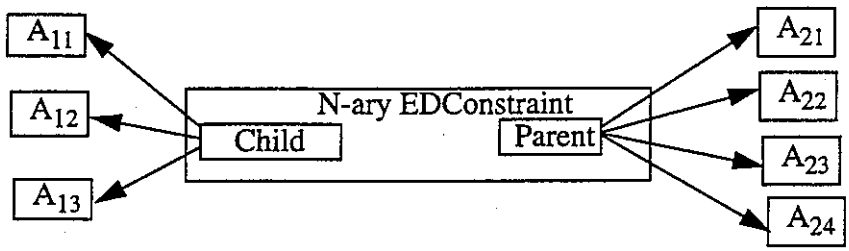
```
A <-> B.
```

The implication of such logical relationship is that, the propagation of PEX value through the binary EDConstraint has to make sure that the same PEX value is maintained between the two activities constrained by a binary EDConstraint.

## 3. 7 Design of the N-ary EDConstraint

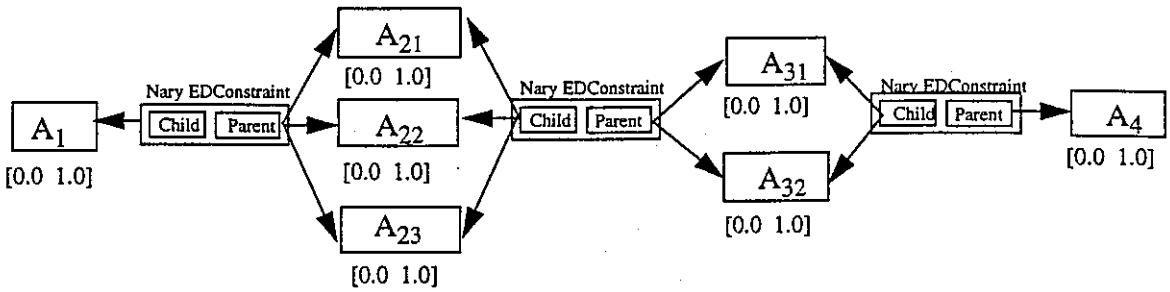
In order to generalize the discussion about these constraints, we will consider a case where an EDConstraint links *n* number of activities to *m* number of activities (where both *n* and *m* >= 1). The structure of the EDConstraint is very much similar to that of the N-ary temporal constraint. As shown in the figure 3.14, the constraint points to two lists of activities - 'Child' and 'Parent'. These two lists represents the activities that are constrained by this constraint.

**Figure 3.14. Structure of the N-ary EDConstraint**




Existence of the list of activities pointed to by the "Child" depends on the existence of the list activities pointed to by the "Parent". Let's now discuss the algorithm for propagating through the N-ary EDConstraint (NEDC). We will work with a simple example of disjunctive activity network as shown below. Figure 3.15 shows that we have three NEDCs between activities $A_1$ and $\{A_2\}$, between $\{A_2\}$ and $\{A_3\}$ and between the activities $\{A_3\}$ and $A_4$. The PEX values of the corresponding activities are shown in the square brackets. These are the values after the initial instantiation of the network and *before* any propagation was done. The EDConstraint specify that activities $\{A_3\}$ depends on the activity $A_4$, activities $\{A_2\}$ depends on the activities $\{A_3\}$ and the activity $A_1$ depends on the activities $\{A_2\}$.

Figure 3.15. An activity network with N-ary EDConstraints (before propagation)




We will use the following notations to describe the propagation algorithms for N-ary EDConstraints.

Figure 3.16. List of notations for the algorithm

$Act_{child}$ = List of activities to which the propagation is following

$Act_{parent}$ = List of activities from which the propagation is coming

$Act_o$ = activity from which the propagation comes;

$PEX_{min}$ = Minimum PEX value of an activity

$PEX_{max}$ = Maximum PEX value of an activity

$PXO_{min}$ = Minimum PEX value of the activity from which the propagation comes.

$PXO_{max}$ = Maximum PEX value of the activity from which the propagation comes.

$num_{child}$ = Number of activities in $Act_{child}$

$num_{parent}$ = Number of activities in $Act_{parent}$

$Alt_o$ = List of alternatives of the activity $Act_o$

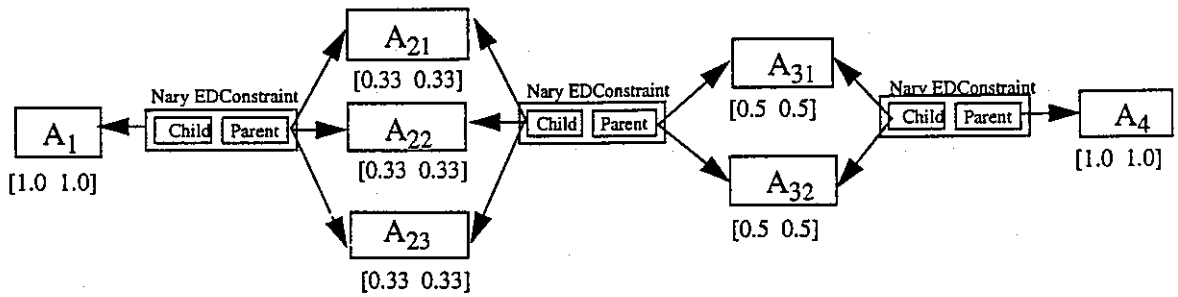
## 3. 7. 1 Full PEX propagation

The full PEX propagation propagates values through the Nary EDConstraint. It guarantees full arc consistency with respect to PEX values regardless of the state of the consistency graph before the propagation. Let's consider that, we begin our forward propagation from activity $A_4$ of figure 3.15. Let us assume that we positively want the activity $A_4$ in the final solution (an example of such activity is the final activity in an activity network). Thus we can set the PEX of $A_4$ to be equal to 1.0 by means of a Unary EDConstraint. The resulting domain of PEX value of $A_4$ will be (1.0, 1.0). Let's now continue propagation from $A_4$ to its dependent activities. Since $A_4$ is the only activity in the 'From' list of the constraint, we will equally distribute

the PEX value of $A_4$ among the activities in $\{A_3\}$. Thus we set the maximum and minimum values of PEX of $A_{31}$, and $A_{32}$ to be equal to 0.5. The reason for this is, initially we are assuming that, all these two activities are equally probable to be executed. We then continue the propagation to activity $\{A_2\}$. As the 'From' list of the N-ary constraint between $\{A_2\}$ and $\{A_3\}$ now contains more than one activity, we first add the maximum and minimum values of PEX for the activities $A_{31}$ and $A_{32}$. We then distribute both the minimum and the maximum values equally among the activities $A_{21}$, $A_{22}$, $A_{23}$. Continuing the propagation to $A_1$, we first sum the maximum and minimum values of PEX for the activities in $\{A_2\}$ and set the PEX of $A_1$ to be equal to 1.0 (we don't need to divide the values as there is no alternative to activity $A_1$). For We can now summarize the above algorithm for full PEX propagation as follows:

```
min = 0.0;
max = 0.0;

if the number of activities in Act_parent = 1
{
        min = PXO_min;
        max = PXO_max;
}
else
{
    for all activities in Act_parent
    {
         min += PEX_min;
         max += PXO_max;
   }
}
for all activities in Act_child
{
          PEX_min = min / num_child;
          PEX_max = max / num_child;
          Propagate the value to downstream activities;
}
```

**Figure 3.17. An activity network with N-ary EDConstraints (after propagation)**

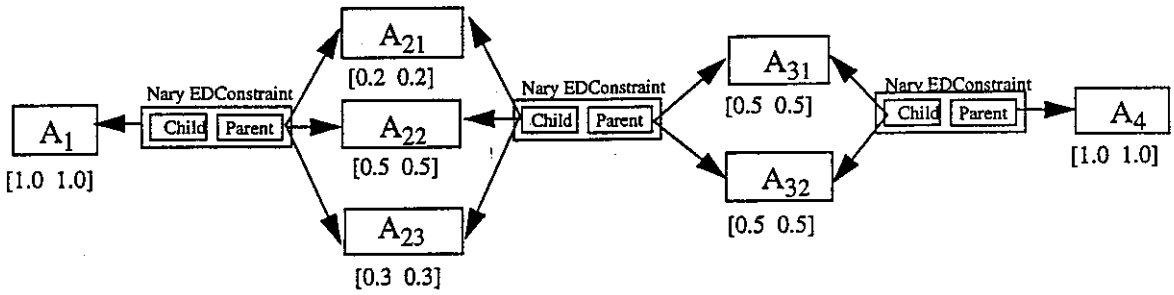


The network assumes the form shown in figure 3.17 after achieving the full arc-consistency (with respect to PEX values):

## 3. 7. 2 PEX propagation

If full PEX arc-B-consistency with respect to the PEX value was held before the assertion of a new commitment, the *PEX propagation* guarantees full PEX arc-B- consistency after this propagation has been performed. We call this function to achieve arc consistency when a PEX value of an activity of in an arc-consistent network has been changed or modified. PEX propagation reestablishes the arc consistency. We will demonstrate the algorithm with an example. Let's consider that we have an activity network shown in figure 3.18 which is fully arc consistent with respect to its PEX value. Let's assume that the PEX value of the activity $A_{22}$

**Figure 3.18. An activity network with N-ary EDConstraints (after full propagation)**




has been modified to (0.4, 0.4) as a result of a commitment. The effect of such change will be contained in the activity set $\{A_2\}$ as the activity $A_{22}$ has alternatives, and will not be propagated to the parent or the child activities of $\{A_2\}$. Thus the activities that will be affected by the resulting PEX propagation are activities $A_{21}$ and $A_{23}$. No other activities in the network will have any effect because of this change. However, if the activity $A_{22}$ had no other alternatives then the propagation will effect all the parent and child activities of $A_{22}$, if it had any. Similarly, if we had other activities dependent on activity $A_{23}$ (e.g. an activity connected to it by means of a binary EDConstraint), we would then have to propagate the effect of this change (change of $A_{23}$ as a result of change of $A_{22}$) to the dependent activities also.

To update the values of $A_{21}$ and $A_{23}$, we first get the sum of three activities before changes were made. The sum for this case is 1.0 (0.2+0.5+0.3). Next we subtract the newly assigned value of $A_{22}$ from the sum. Which is 1.0 - 0.4 = 0.6. We then distribute this value to activities $A_{21}$ and $A_{23}$ in the same ratio as it was before. Thus for this case we will assign a value of 0.6

* 0.2 /(0.2 + 0.3) = 0.24 to activity $A_{21}$ and a value of 0.6 * 0.3 /(0.2 + 0.3) = 0.36 to the activity $A_{23}$. The rationale behind this distribution is that, unless we have any reason to modify the distribution of PEX among the alternative activities, we would continue to respect the ratio at which the PEX was distributed among them. We can now specify our algorithm as follows (using the same notation):

**Forward Propagation:**

```
min     = 0.0;
max     = 0.0;

if the number of activities in ACT_parent = 1
{
      for all activities in ACT_child
      {
        PEX_min = PXO_min / NUM_child;
        PEX_max = PXO_max / NUM_child;

        Propagate the value to the dependent activities;
      }
}
else if the number of alternatives of ACT_parent > 1
{

   TOTPEX_min = sum of PEX_min of all activities in the ACT_parent
                before changes were made to Act_o;
   TOTPEX_max = sum of PEX_max of all activities in the ACT_parent
                before changes were made to Act_o;

   REPEX_min = TOTPEX_min- PXO_min;
   REPEX_max = TOTPEX_max - PXO_max;

    for all activities in ACT_parent
    {
         min += PEX_min;
         max += PEX_max;
    }
    for all activities in ACT_parent
    {
         TEMP_min = REPEX_min * (PEX_min / min);
         PEX_min = TEMP_min;
         TEMP_max = REPEX_max * (PEX_max / max);
         PEX_max = TEMP_max;
         Propagate the value to the dependent activities;
    }
}
```

**Backward Propagation:**

```
min     = 0.0;
```

```
max     = 0.0;

if the number of activities in ACT_child = 1
{
      for all activities in ACT_parent
      {
        PEX_min = PXO_min / NUM_parent;
        PEX_max = PXO_max / NUM_parent;

        Propagate the value to the parent activities;
      }
}
else if the number of alternatives of ACT_child > 1
{

   TOTPEX_min = sum of PEX_min of all activities in the ACT_child
                before changes were made to Act_o;
   TOTPEX_max = sum of PEX_max of all activities in the ACT_parent
                before changes were made to Act_o;

   REPEX_min = TOTPEX_min- PXO_min;
   REPEX_max = TOTPEX_max - PXO_max;

    for all activities in ACT_child
    {
         min += PEX_min;
         max += PEX_max;
    }
    for all activities in ACT_child
    {
         TEMP_min = REPEX_min * (PEX_min / min);
         PEX_min = TEMP_min;
         TEMP_max = REPEX_max * (PEX_max / max);
         PEX_max = TEMP_max;
         Propagate the value to the parent activities;
    }
}
```

With these representation frameworks we will now define the alternative scheduling problem in the next section.

## 3. 8 Problem Definition

The scheduling problem described here is a variation of the standard job-shop scheduling problem described in the literature. The main distinguishing feature of this problem is that, it involves alternatives (disjunctiveness). We tend to view this problem in the class of disjunctive scheduling problems.

## 3. 8. 1 Problem Components

**Job**: The main component of the problem is set of jobs, $J = \{J_1, J_2 \ldots\ldots J_n\}$. Jobs may or may not have individual start and due dates. In cases where they don't have, their start and due dates are constrained by the scheduling horizon. The scheduling horizon is bounded by two time points - the earliest start time, $et_j$ and the due date, $dd_j$. These time bounds specify that no job can be started before the earliest start time and no job can be finished at a time later than the due date. Each job also specifies a process plan or process routing. The process plan defines the sequence of activities along with the alternatives that needs to be executed in order successfully process the job. No explicit temporal relationship exists between the jobs, i.e., the jobs are independent of one another.

**Activity:** Each job $J_i$ comprises of a set of activities $A_i = \{A_{i1}, A_{i2} \ldots\ldots A_{in}\}$. In other words, each activity belongs to a certain job. There are temporal relationships between the activities within a particular job. Activities are specified by the following three

integer interval variables:

$dur_{ijk}$ = duration of activity i of job j on resource, $r_k$, etj < $dur_{ijk}$ <= $dd_j$
$est_{ij}$ = earliest start time of activity i of job j, $et_j$ < $est_{ij}$ <= $dd_j$
$lft_{ij}$ = latest finish time of activity i of job j, $et_j$ < $lft_{ij}$ <= $dd_j$

and a real interval variable:

$pex_{ij}$ = PEX of activity i of job j, 0.0 < $pex_{ij}$ <= 1.0

Each activity can have several alternatives. $A_{ij} = \{A1_{ij}\ A2_{ij} \ldots\ldots Al_{ij}\}$. The alternatives are of type XOR, i.e., only one of the alternatives from the set is going to be executed. The activities also specifies their resource requirements. Each activity can have a set of alternative resources to choose from. Thus, $R_{ij} = \{R1_{ij}, R2_{ij} \ldots\ldots Rm_{ij}\}$, specifies the set of alternative resources required by activity $A_{ij}$. Like activity alternatives, the resources alternatives are also of type XOR. We are assuming that each activity requires only one unit of a particular resource.

**Resource:** Resources are required by activities for their execution. The resources that are used for the current version of the problem are all "machine" resources. What it means is that, these type of resources are *used*, as opposed to *consumed*, by the activities. Thus an activity returns the full capacity of the resource at the end of its execution. For simplification, we

are assuming that the resources are of unit capacity. Thus we cannot execute more that one activity simultaneously on a single resource.

## 3. 8. 2 Constraints

The following three types of constraints are used in modelling the problem.

### 3. 8. 2. 1 Temporal Constraint

As specified earlier, scheduling horizon imposes temporal constraints on the jobs and in turn to all the activities belonging to these jobs. In addition to that, the activities within a job are temporally constrained by their sequence in the process routing. The process routings specifies that certain activity cannot be started before/after the finish/start of other activity. Thus if an activity $A_{ij}$ must precede an activity $A_{oj}$ of the same job J, then the constraint specifies that:

$$est_{ij} + dur_{ij} <= est_{oj} \quad \text{and}$$
$$lft_{ij} + dur_{ij} <= lft_{oj}$$

Three types of temporal constraints are specified. Namely, unary, binary and n-ary. Unary temporal constraints constrain temporal interval variables (start time, duration and end time) associated with each activity. Binary constraints are used to establish temporal relationship between two activities and N-ary constraints are used to establish temporal relationship between two sets of disjunctive activities.

### 3. 8. 2. 2 Resource Constraint

The resource constraint restricts the number of activities that can be executed simultaneously on a particular resource. Given the unit capacity resource and unit demand from the activities, this constraint specifies that no more than one activity can be executed simultaneously on a single resource. This constraints can be represented in a disjunctive form. Thus if two activities $A_{ij}$ and $A_{pq}$ uses the same resource, then

$$\text{either,} \quad est_{ij} + dur_{ij} <= est_{pq}$$
$$\text{or,} \quad est_{pq} + dur_{pq} <= est_{ij}$$

### 3. 8. 2. 3 Existence Dependency Constraint (EDConstraint)

These constraints specified the existence of one activity on the other. Thus if an activity $A_{ij}$ depends on another activity $A_{pq}$, this means that if at any point in the search process activity $A_{pq}$ cease to exist, activity $A_{ij}$ would also cease to exist. Like temporal constraints, three types

of EDConstraint are specified. The unary EDConstraint restricts the value of PEX of an activity. The binary EDConstraint establishes dependency relationship between two activities and the N-ary EDConstraint established dependency relationship between two sets of alternative activities.

As with the any other class of job-shop scheduling problem, the goal is to find a feasible schedule for the jobs without violating any of the constraints.

## 3. 9 Summary

This chapter demonstrated our approach to model and represent alternative scheduling problem. We started with a simple problem of alternative resources. Our approach involves converting alternative resource problem into alternative activity problem. We introduced N-ary temporal constraints and associated propagation algorithm for binding alternative activities in the constraint network. We also introduced the notion of probability of existence (PEX) of an activity. To facilitate the propagation of PEX variable through the network and to express the dependency relationship among the activities, we introduced a new type of constraint called, Existence Dependency Constraint (EDConstraint). Propagation algorithm for these new types of constraints are also specified. Our modeling and representation framework is generic enough across a wide range of alternative scheduling problems. In addition to the alternative resource and alternative activity problems, our framework can be also be used for representing alternative process plan problem. The example of such representation is given in Chapter 6 of this report.

# Chapter 4 Solving Alternative Scheduling Problem

This chapter describes our approach towards solving alternative scheduling problem as a constraint satisfaction problem. We intend to incorporate our methodology in ODO's problem solving model. For this, we will first give an overview of ODO, a constraint based scheduling system being developed at the Enterprise Integration Laboratory of the University of Toronto. We will summarize the main features of ODO in order to create a context for the current work. For a detail description of ODO's constraint model and problem solving methodology, please refer to [Davis 94]. We then present the extension to the current model to solve our problem. As we will see in later sections, extensions were done both in problem representation and problem solving methodology.

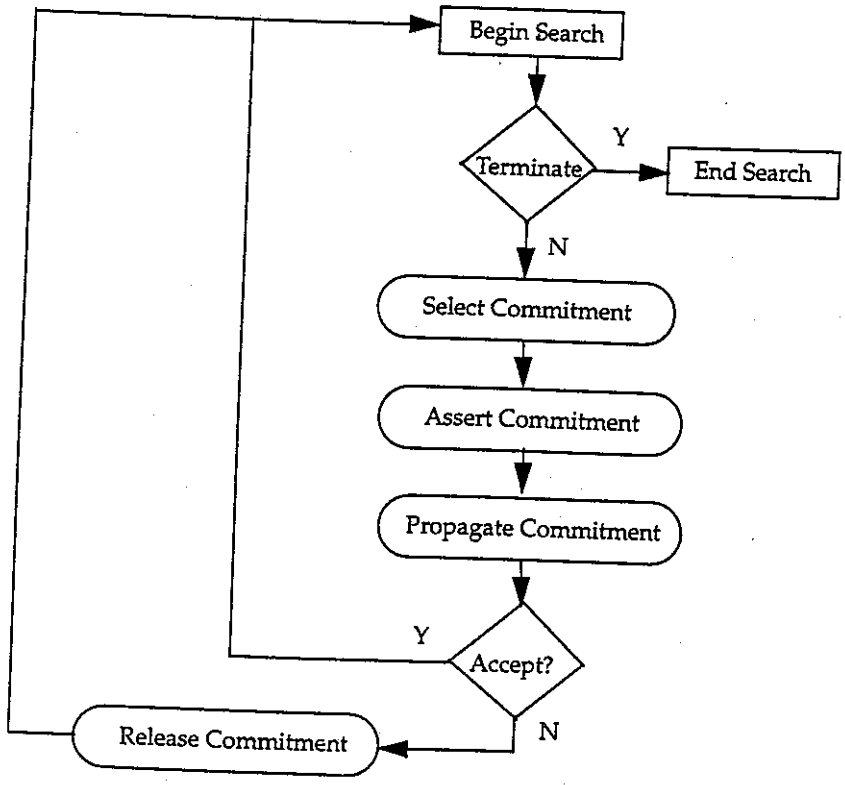
## 4.1 ODO: A Unified Model for Constraint Based Scheduling

Research in constraint-based scheduling is initiated with the premise that scheduling problem lends itself to the paradigm of a constraint satisfaction problem (CSP). The constraint based problem solving approach provides a generic representation and problem solving methods for solving constrained problems. Within the CSP framework, a problem is represented by a constraint graph, with nodes and arcs connecting the nodes [Mackworth 86]. The nodes correspond to the variables of the problems and arcs to the constraints. Each variable has a domains. The solution of the problem involves assigning values to the variables from their respective domains such that none of the constraint are violated. For an excellent introductory overview of CSP techniques and tools please refer to [Kumar 92]. For a detail description of how a CSP framework can be used to represent scheduling problems please refer to [Davis 94].

In addition to the issue of representation, the other motivation for adopting CSP framework for scheduling problem includes "the availability of problem solving tools and the ability to exploit problem structure" [Davis 94]. In recent years there has been an explosive number of growth in research in constraint-based scheduling. One can come up with several groupings or taxonomies to classify these research efforts. A very prominent distinction is that one

based on search methods used in the system. These are constructive and repair based search. In constructive search, one starts with an unassigned constraint graph and incrementally assigns values to the variables such that none of the constraints are violated. If the graph reaches a state where no consistent value is found in the domain of a variable then it backtracks to the previous state and assigns a different value. The process continues until all the variables are assigned consistent values, or any other stopping criterion is met. With repair based approach, one starts with a constraint network where all the variables are assigned regardless of their consistency. Then the search proceeds by repairing the inconsistent assignments. If a dead-end is reached then the search backtracks. The process continues until all the inconsistent assignments are repaired or any other stopping criterion is met. The objective of the research in ODO is to provide a unified problem solving model that captures both of these two paradigms, so that the relationship between the search methods and the problem structure can be exploited.

**Figure 4.1. Unified model for constraint-based scheduling [Davis 94]**



## 4. 2 Search as Commitment Transformation

The unified model for constraint-based scheduling is based on the observation that search process in the constraint network can be modeled as the assertion and retraction of commitments. The commitment refers to those decisions that change the search states. Thus, the commitment may be assigning a variable to the value (in constructive search), posting a temporal constraint, or repairing an inconsistent assignment (repair based search). The search proceeds by posting new commitments, propagating the effect of the commitment, evaluating the resulting state, if it is acceptable then proceeding further to make a new commitment. If it is not feasible, then retracts the commitment (backtracks). With this generalization of search process [Davis 94] proposed the search loop shown in figure 4.1 as a representative of the unified model and incorporated it in a constraint-based scheduling system called ODO.

## 4. 3 Problem solving model of ODO

The select commitment stage in ODO comprises the variable and the value selection. The heuristics used for these are described in the following two sections. Like any other constraint based system ODO's problem solving model includes variable and value selection stage. The following sections describe these steps in detail.

### 4. 3. 1 Variable Selection Heuristic

The variable selection heuristic of ODO is based upon the work done in MicroBOSS [Sadeh 91]. MicroBOSS uses the problem solving model of Constraint Heuristic Search (CHS) [Fox 89]. CHS uses the properties of the constraint graph which are called "textures" to guide the search process. The variable and value selection heuristics is based on two such texture measurements. These are 'reliance' and 'contention'. MicroBOSS also introduced the notion of micro opportunistic search. The focal point of each decision is an individual activity and the opportunism in search is introduced by its ability to dynamically track the most critical resource. The current version of ODO is based on these principles, although some extensions are made in the variable and value section heuristics. These would be discussed in the appropriate sections. We will be using the following notations throughout the chapter for our discussion.

$A_i$ = Activity i

$R_j$ = Resource j

$Area_i$ = Area under demand curve of activity i

$P_j$ = Probability of breakage of the maximum capacity constraint of resource j

$T_j$ = Time point on $R_j$ where the probability of breakage of the maximum constraint is $P_j$

t = Time point referring to the individual demand curve of an activity

ADC = Aggregate Demand Curve of a resource

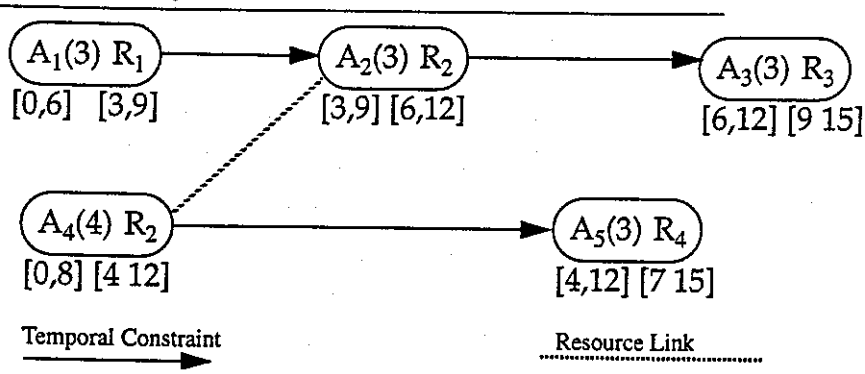
PBMCC = Probability of Breakage of the Maximum Capacity Constraint

APV = Assign PEX Value (commitment)

PEX = Probability of EXistence

For the sake of explanation we will use a simple example of a scheduling problem. The problem, as shown in the figure 4.2, consists of a network of five activities. The figure shows the network after an initial consistency labelling (we assumed that the scheduling horizon is given to be [0, 15]).

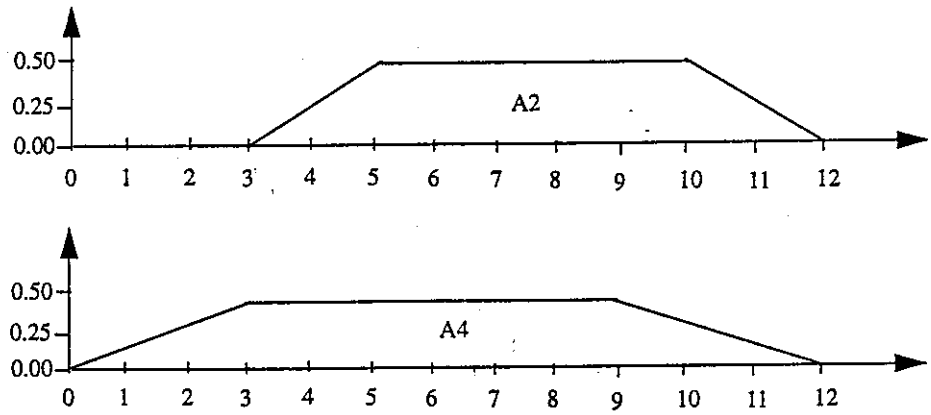
**Figure 4.2. A simple activity network with five activities**




Activity names are specified along with their durations and resource requirements. Thus activity $A_1$ has a duration of 3 time units and uses resource $R_1$. The two numbers within the first square bracket define the EST and LST and the numbers in the second square bracket define the EET and LET of the corresponding activities. Temporal constraints specify the temporal dependency (e.g. $A_4$ must precede $A_5$). If two activities use the same resource then a resource link is established between them (e.g. $A_2$ and $A_5$). After achieving the initial arc consistency, the algorithm works as follows:

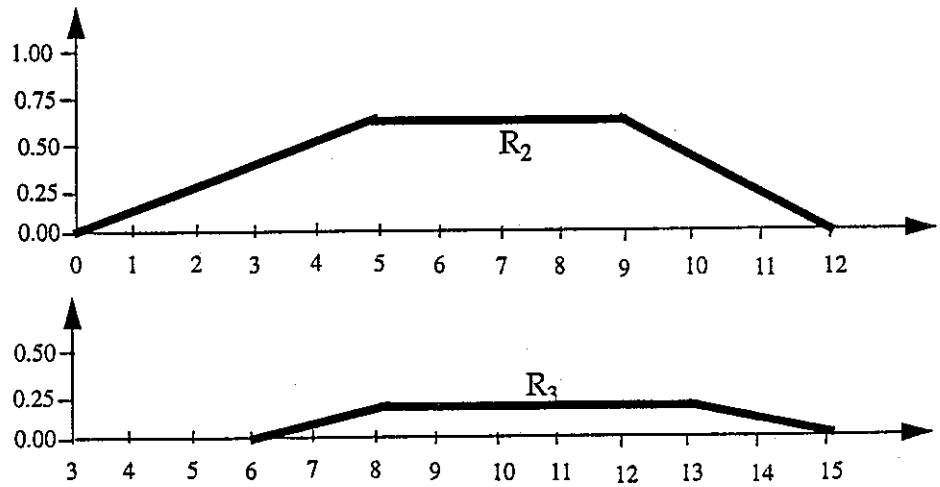
- Generate a probabilistic activity demand curves corresponding to each activity in the network. These curves indicate the probability that an activity will use the resource at any given time points. The curves are calculated on the basis of uniform probability distribution of the possible start times of the associated activity. For detail description and example of how to generate these curves, please refer to [Sadeh 91]. This demand based texture measurement is called 'reliance'. Figure 4.3 shows some examples of such curves.

Figure 4.3. **Individual demand curves of activities $A_2$ and $A_4$**




- These activity demand curves are then algebraically added to generate aggregate demand curve (ADC) corresponding to each resource. ADC curve gives a measure of resource contention.

Figure 4.4. **Aggregate Demand Curves of resources $R_2$ and $R_3$**



- For each of the ADCs of figure 4.4, estimate the probability of breakage of the *maximum capacity* constraint(PBMCC). The maximum capacity for the scheduling problem refers to the capacity of individual machines. The PBMCC is a generalization of contention measure proposed by [Sadeh 91]. It estimates the probability of maximum capacity at each time points and returns the time point where the PBMCC is the highest. For detail explanation on how to calculate these values please refer to [Beck 97]. Let's assume that we have a value of $[P_2, T_2]$ for resource $R_2$, and $[P_3, T_3]$ for resource $R_3$, where $P_i$ is the PBMC on resource $R_i$ at time point $T_i$.
- Choose the resource having the largest $P_i$. Let it be $R_2$. Identify the activities on these resource ($R_2$) that are contributing to resource contention at the critical time point $T_2$. e.g. $A_2$, $A_4$
- Order the activities in descending order in relation to their reliance placed on the critical resource at the critical time point. For the present case, the list would be $[A_2, A_4]$.
- Identify the first two activities from the list that have no temporal path between them. For the example in hand, these would be $A_4$ and $A_2$. The reason that we make the check for temporal path is to prevent cycling. Unlike [Sadeh 91]'s variable selection heuristic we consider two most contending activities instead of just one.

This is the end of our variable selection step. We then move on to the next step of the search process, which is called value selection.

## 4. 3. 2 Value Selection

Once we have identified the two most reliant activities (i.e. variables), the next step is to decide which value to assign to these variables. The objective of the value assignment is to reduce the contention of the two activities at the critical time point or in our measure, reducing the probability of breakage of the maximum capacity constraint at the most critical time point. Instead of assigning start times (as it was done in MicroBOSS) to the most critical activity, our value selection posts a sequencing constraint between the two most contending activities. The motivation behind this is by ensuring that the start times of these two activities do not overlap in time, we can reduce the contention at the critical time point. This means that, one of these activities is going to precede the other. This leaves us with two alternatives to choose from, and the goal of our value selection is to decide which value to pick, i.e. to decide which activity to precede the other.

The advantage of posting such sequencing constraint is that, instead of getting a single feasible solution, it gives us a family of feasible solutions. This is a least-commitment approach compared to start time assignments [Muscettola 94]. The advantage of such approach lies in its flexibility to react to the unanticipated events. Uncertainties in a manufacturing environments are norm rather than exception - machines break down, material shortage occur, workers fail to show-up. The effect of such unplanned events can be absorbed to a certain degree, if we have a time window instead of a time points associated with the scheduled activity.

In order to decide which value to pick (i.e. which precedence constraint to post) we used the CBA(Constraint Based Analysis) algorithm as proposed by [Ercshler 76]. CBA decides about the temporal ordering of two activities that are contending for same resource in an overlapping time interval. Note that, this is an exact algorithm, not a heuristic. Thus we don't need to be concerned about the survivability of the decision given by CBA. The following section provides an overview of the CBA algorithm.

## 4. 3. 3 Constraint Based Analysis

Let's consider that we have two activities, A and B having an overlapping time interval which are contending for the same resource. The EST, LET and the duration of these two activities are $est_A$, $let_A$, $dur_A$, and $est_B$, $let_B$, $dur_B$ respectively. Let,

$$t_A = let_A - est_B$$
$$t_B = let_B - est_A$$
$$t_C = dur_A + dur_B$$

Now,

Case 1: If
$t_A < t_C \leq t_B$

then activity A should precede activity B

Case 2: If
$t_B < t_C \leq t_A$
then activity B should precede activity A

Case 3: If
$t_C > t_B$
and
$t_C > t_A$
then no feasible solution exists.

Case 4: If
$t_C \leq t_B$

and

$$t_C \leq t_A$$

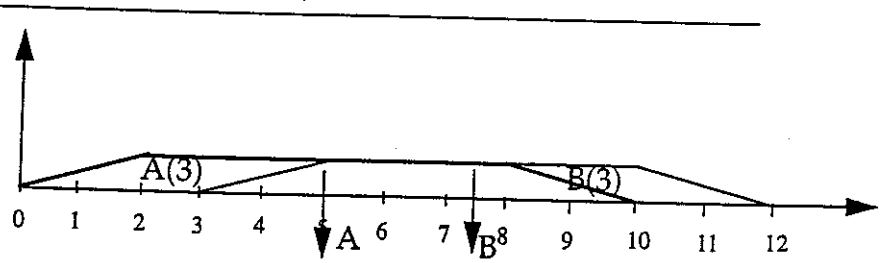
then either sequencing is possible.

It is the last case that makes the point for further research. As there is no exact algorithm to decide about this fourth case, research efforts are directed towards the development of heuristic measures that help to decide between the alternative sequencing decisions. Two such promising heuristics, that have been reported to have achieved good results are 'Centroid Based Heuristic' and 'Slack Based Heuristic'. These heuristics are described in the following sections.

## 4. 3. 4 Centroid Based Heuristic

The centroid heuristic specifies the ordering for the two activities based on the horizontal axis of the centroid of the activity's individual demand curve [Muscettola 94]. The heuristic specifies that if the horizontal axis of the centroid of an activity's demand curve precedes to that of the other, then the former should precede the latter. Let's consider that we have two activities A and B, each having a duration of 3. Their EST and LET are shown in the figure 4.5. Applying CBA algorithm, we see that either ordering is possible. Applying the centroid heuristic we see that the horizontal axis of the centroid of the demand curve of A precedes the horizontal axis of the centroid of the activity demand curve of B (figure 4.5).

**Figure 4.5. Centroid based heuristic**



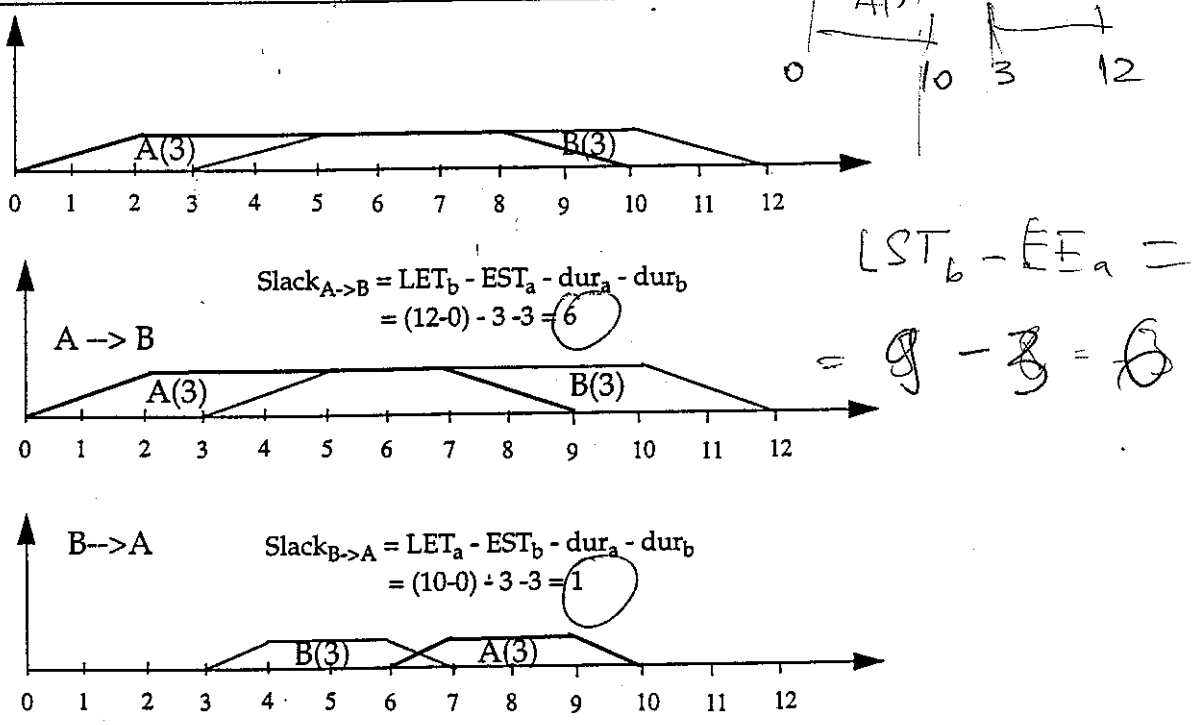

Thus activity A must precede activity B. [Muscettola 94] has reported to have achieved good results with the use of such heuristics.

## 4. 3. 5 Slack-based heuristic

Another heuristic for situations like the one we have in Case 4 of CBA in section 4.3.3, is called slack based heuristic. This heuristic is first proposed by [Smith 93]. The attractiveness of this heuristic lies in its simplicity. [Smith 93] reported a strong result on a set of benchmark

problems. They later extended this heuristic to a more general form [Cheng 95]. The heuristic is as follows. Let's consider two activities A and B contending for a critical resource and have a overlapping time interval. The heuristic says that whatever decision leaves the most temporal slack should be selected. The motivation behind the choice is that, by choosing the ordering corresponding to higher slack, we are not over constraining the problem space and leaving room for future value assignments. Let's consider that we have two activities A and B, each having a duration of 3. Their EST and LET are shown in the figure 4.6. Applying CBA, algorithm, we see that either ordering is possible for this case. This prompted us to apply the slack-based heuristic. As shown in the figure 4.6, the ordering of A->B leaves most slack and is selected to be the preferred ordering.

**Figure 4.6. Slack Based Heuristic**




## 4. 4 Extension to ODO's problem solving model for alternative scheduling

From the definition of the problem specified in the last chapter, it is evident that the problem that we are trying to solve is different from the standard job-shop scheduling problem. The existence of alternatives in the constraint graph introduces another dimension of complexity and needs extension to many of the problem solving modules of ODO. Moreover, our exten-

sion to the problem representation needs to be incorporated in order to solve the alternative scheduling problems. We have identified the following nine areas where we need to work in order to solve our problem with ODO:

- variable selection
- value selection
- new texture measurement for selecting the best commitment from a set of alte natives.
- new commitment for updating or modifying the PEX value of the activity on the basis of the texture measurements.
- pruning alternative choices
- dynamically adding or removing activities to the constraint network.
- infeasibility/dead end checks
- termination criteria
- problem description language

We will deal with each of these in the subsequent sections. Please note that, our discussion is based on the assumption that our activity network contains disjunctive nodes *only of type XOR*. In order to explain these extensions and modifications, we will work with a simple example. We will take the example given in figure 4.2 and introduce an alternative activity to the network. With this modification our network assumes the form shown in figure 4.7.

**Figure 4.7. A simple activity network with an alternative activity**

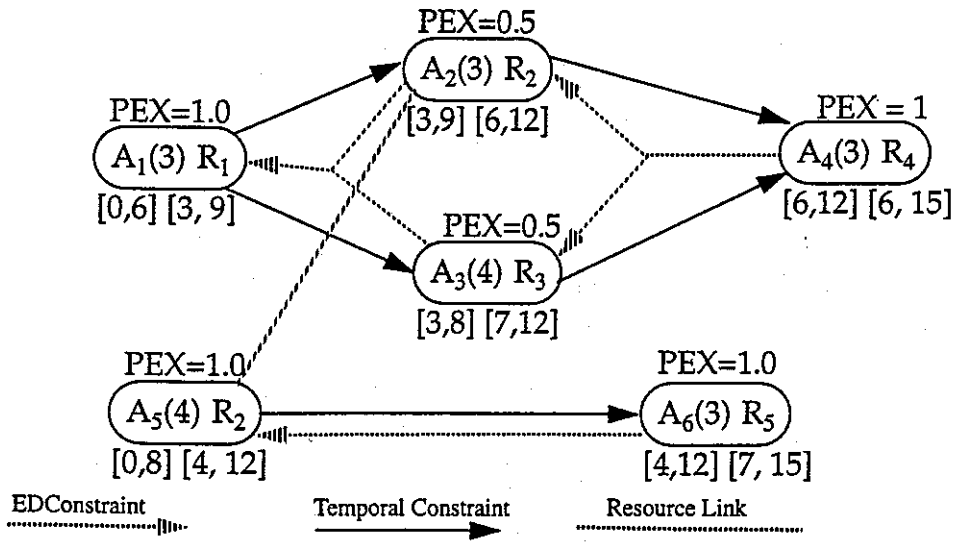


Figure 4.7 specifies a network of six activities after an initial consistency labelling (we assumed that the scheduling horizon is given to be [0, 15]). The difference between this example and the one given earlier is that, here we have an alternative activity for $A_2$. Also we have introduced PEX values and dependency constraints to the constraint network. Temporal constraints specify the temporal dependency (e.g. $A_5$ must precede $A_6$) and the EDConstraints specify the existence dependency (e.g. existence of $A_6$ depends on the existence of $A_5$). The N-ary temporal constraints are also introduced between activities $A_1$ and $A_2$, $A_3$ and also between activity $A_2$, $A_3$ and $A_4$. In addition to temporal propagation, we also have PEX propagation. The PEX propagation starts propagating from activity $A_4$. By default it has a PEX value of 1.0. By propagating this value to $A_2$ and $A_3$ we set PEX value of 0.5 to each of these activities. Similarly we set the PEX of $A_1$ by propagating it from $A_2$ and $A_3$. All of our subsequent discussion will be based on this activity network (figure 4.7).

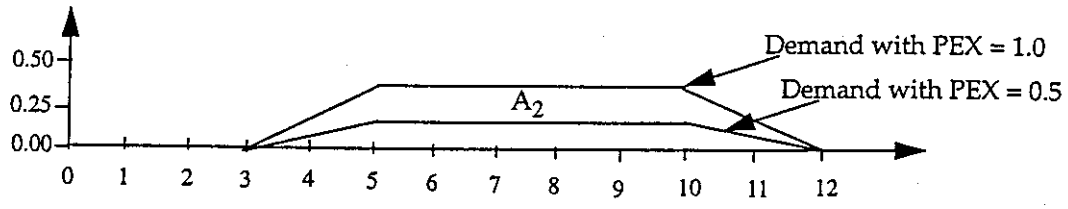
## 4. 5 Modification in Variable Selection Heuristic

Two changes were made in our variable selection heuristic. The following sections describe these changes.

### 4. 5. 1 Changes in Texture Measure for Variable Selection

Recall that, in our variable selection heuristic, we constructed activity demand curves corresponding to each activity. These demand curves were aggregated for each resource to find out the most contended for resource. If we follow the same approach to construct the demand curves for the alternative scheduling problem, then we would be overconstraining the estimate. The reliance/contention based texture measurements is based on the assumption that each activity in the activity network is going to be executed in order to complete the schedule. But the introduction of the PEX variable to our activity overrides this assumption. Now, our texture measurements have to deal with activities whose probability of existence (PEX) varies between 0< = PEX <=1.0. Which means that, some of the activities may not exist in the final solution. This would definitely have some impact on the probabilistic demand curves that we compute for our texture measurements. One obvious way to reflect this probability on demand is to multiply the demand of the activity with it's PEX value. Thus if an activity is demanding one unit of resource $R_1$, and has a PEX value of 0.5, then the probability of demand that activity places on $R_1$ would be 0.5 instead of 1.0. The effect of such change is demonstrated in the figure 4.8.

Figure 4.8. Modified demand curve of activity $A_2$ for PEX = 0.5




Since our activity demand curve reflects probabilistic demand, not the actual demand, multiplying the demand with the probability of existence is a valid measure of the probabilistic demand.

## 4. 5. 2 Changes in Variable Selection

In our variable selection heuristic, after constructing the aggregate demand curve for each resource, we calculated the probability of breakage of maximum capacity constraint (PBMCC). We identified the two activities that contributed most in the demand at the critical time point on the critical resource.

Our objective was to post sequencing constraint between them in order to reduce contention on the resource at the critical time point. The introduction of alternative activity opens up another way of reducing the contention at the critical time point. If any of the activities on the critical resource has an alternative activity (either because of the existence of an alternative resource or because of the existence of alternative activity), we can decide not to execute that activity, thereby reducing the contention on the critical resource. Let us consider that the critical resource for the activity network of figure 4.7 is $R_2$ and the two most reliant activities are $A_2$ and $A_5$. In addition to posting a sequencing constraint between $A_2$ and $A_5$, the other way we can reduce the contention is by deciding not to execute activity $A_2$. This is a valid choice as long as we have an alternative for $A_2$. For our example, we have the activity $A_3$, which is an alternative of $A_2$. Thus we can consider the alternative of not executing $A_2$. This shows that we need to modify our variable selection heuristic in order to include the other alternative. Our modified variable selection heuristic is as follows: (references are made to the activities of figure 4.7)

- Generate probabilistic activity demand curve corresponding to each individual activity in the network.
- Generate aggregate demand curves (ADC) corresponding to each resource by algebraically adding the individual activity demand curves.

- For each of these ADCs, estimate the probability of breakage of the *maximum capacity* constraint (PBMCC). Let's assume that we have a value of $[T_2, P_2]$ for resource $R_2$, and $[T_3, P_3]$ for resource $R_3$, where $P_i$ is the PBMC on resource $R_i$ at time point $T_i$.
- Choose the resource having the largest $P_i$. Let it be $R_2$. Identify the activities on this resource ($R_2$) that are contributing to resource contention at the critical time point $T_2$. e.g. $A_2$, $A_5$.
- Order the activities in descending order in relation to their demand placed on the critical resource at the critical time point. For this case, the list would be something like $[A_2, A_5, \ldots\ldots]$.
- Identify the first two activities from the list that have no temporal path between them. For the example in hand we pick up $A_4$ and $A_2$.
- Also identify an activity (scan from top to bottom) from the list having an alternative(s). In this case it would be activity $A_2$.

## 4. 6 Modification of Value Selection

The value selection heuristic for the current ODO works in two phases. At first it uses CBA algorithm for choosing the preferred ordering between the two most contending activities identified at the variable selection stage. As mentioned earlier, CBA is an exact algorithm and we want to use this algorithm first before trying any other heuristics. If CBA fails to give us a decision we move into second phase and use heuristics to decide the preferred ordering. Changes were made in both of these phases. The following sections describe these changes.

### 4. 6. 1 Changes in the CBA Algorithm

At a first thought it might appear that, since CBA decides about posting sequencing constraint between the *two* most contending activities, the introduction of the alternatives would not have any impact on this algorithm. But looking back at the CBA algorithm, we see that for Case 3, the CBA algorithm decides that because of the temporal ordering of the two activities, no ordering is possible and thereby no feasible solution exists for the problem. The existence of alternatives may invalidate this decision. If we have an alternative for either of these two activities, then we can get around this problem by not executing one of these two most contending activities. With this modification, our CBA algorithm is as follows:

Let's consider that we have identified the two activities A and B. The EST, LET and the duration of these two activities are $est_A$, $let_A$, $dur_A$, and $est_B$, $let_B$ $dur_B$ respectively. Now if,

$$t_A = let_A - est_B$$
$$t_B = let_B - est_A$$
$$t_C = dur_A + dur_B$$

Now,

Case 1: If
$t_A < t_C \leq t_B$

then activity A should precede activity B

Case 2: If
$t_B < t_C \leq t_A$

then activity B should precede activity A

Case 3: If
$t_C > t_B$
and
$t_C > t_A$

then if (activity A, or both A and B has an alternative activity)
decide not to execute activity A
else if (activity B has an alternative activity)
decide not to execute activity B.
else
no feasible solution exists.

Case 4: If
$t_C \leq t_B$
and
$t_C \leq t_A$

then either sequencing is possible.

Note that, we first check for the alternative of activity A and then for activity B. This ordering is important. Because, our objective is to reduce the contention as much as possible at the critical time point. And since activity A has the highest demand, we prefer not to execute A over not executing B.

## 4. 6. 2 Applicability of the Existing Heuristics

Recall that, when CBA failed to give a decision (Case 4 of CBA), other heuristics were used to find the preferred ordering. These heuristics tend to decide the sequencing commitment between the two variables (two most contending activities) that were identified in the variable selection stage. Thus if we have two most contending activities $A_2$ and $A_4$ (refer to figure 4.7), the heuristic had to pick one of the following two values in order to reduce the contention at the critical time point:

- $A_2$ must precede $A_4$, $A_2$ --> $A_4$
- $A_4$ must precede $A_2$, $A_4$ --> $A_2$

But, for our problem, we have three choices to choose from. If we identify two most contending activities to be $A_2$ and $A_4$, and $A_2$ has an alternative, then our value selection must choose from one of the followings choices (refer to figure 4.7):

- $A_2$ must precede $A_4$, $A_2$ --> $A_4$
- $A_4$ must precede $A_2$, $A_4$ --> $A_2$
- Do not execute $A_2$

In order to choose a value, we need to compare these alternatives by weighing the goodness of each decision. Such measure of goodness should be based on the criterion that, whatever decision we take it should result either in the maximum reduction of contention, or in the minimum increase in contention. The alternative which reduces the contention most (or increase the contention least) should be picked up as our next commitment.

Two of the heuristics that were used in the previous version of ODO for comparing the first two alternatives are, 'Slack based heuristic' and 'Centroid based heuristic'. These heuristics were described earlier. Let us examine whether we can use these heuristics to compare the three alternatives.

The slack based heuristic can discriminate between the changes in the first two choices but it cannot compare the third alternative with the first two. The reason for this is the heuristic calculates the slack values corresponding to the first two choices and bases its decision on these two values (slacks). The ordering which leaves the most slack is chosen. But for the case in hand, if the third alternative activity is different than the first two activities, then the non-existence of the third activity has no impact on the slack values of the other two activities. If the third alternative is one of the first two activities (as in our example), then the slack heuristic would always prefer the third alternative as it would result in the most slack. The heuristic fails to discern among the alternatives.

The centroid heuristic bases its decision on the position of the horizontal axis of the centroid of the activity demand curves. The sequencing decision is based on the existing ordering of the centroidal axes. But in this case, the assertion of third alternative results in the non-existence of an activity and its centroidal axis cannot be compared in relation to other two activi-

ties (or one activity as in our example). Thus, this heuristic also fails to give us a measure to discern among the three values. The failure of these two heuristics prompted us to look for new heuristic measures. Toward this end, we propose the following.

## 4. 7 Recomputation of the PBMCC

### 4. 7. 1 Recomputing PBMCC across all resources

This heuristic is based on the observation that, while selecting a commitment, we should take into account the global effect of such decision. The heuristic recomputes the probability of breakage of all the resources after propagating the effect of a commitment through the entire constraint graph, and then decides upon on the basis of the value of the new probabilities. This is full look-ahead and this would give us an accurate measure for comparing a set of competing alternatives. Let's examine the steps involved in such measurement. For *each alternative* choices (values):

- propagate the effect of the decision through the constrained graph.
- recompute the individual activity demand curve for each affected activity in the network
- recompute the aggregate demand curve corresponding to each affected resource. The reason for recomputing the probability for all the resources is that, as a result of our decision, some other resource may become more critical than the current one. We want to track down such budding criticalities.
- recalculate the probability curve of breakage of the maximum constraint for each resource.
- take the maximum of all the PBMCCs.
- return back to the original state by undoing all the changes in time intervals, activity demand curves, aggregate demand curves and the variance curves (variance curves are used for calculating the probability of breakage of maximum capacity constraint. For explanation on the variance curve please refer to [Beck 97]).
- Identify the commitment that results in the minimum probability of breakage across the entire constraint graph and assert this to be the next commitment.

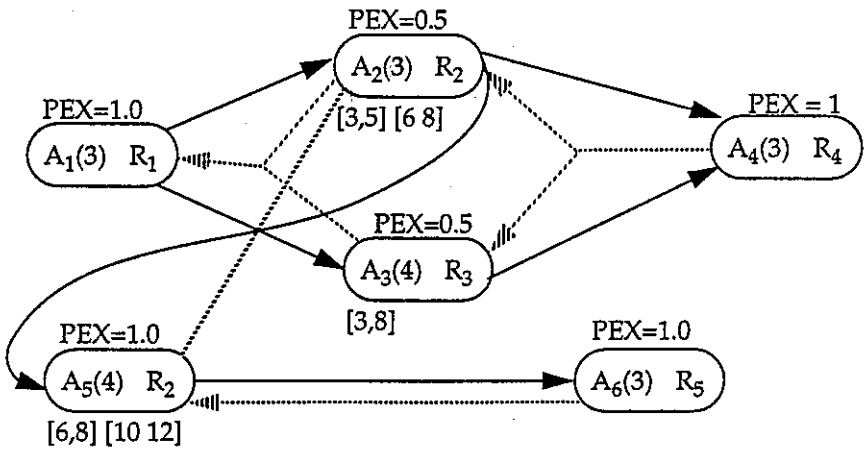
Although this heuristic is a very accurate measure of the goodness of our decision, the computational overhead involved in the process is enormous and it is unrealistic to use the technique even for a small size of an alternative scheduling problem. This has motivated us to

think about a cheaper means of *recomputation* of the probability values. The following heuristic is intended for this.

## 4. 7. 2 Recomputing PBMCC on a subset of resources

Instead of recomputing the probability of breakage of *all* the resources, we can get a fairly good estimate of our decision if we confine our recomputation only to a *subset* of all the resources. For sequencing commitment the subset comprises of only the critical resource. For Assign PEX Value (APV) commitment the subset comprises of current critical resource as well as all the resources that are being used by other alternatives of the concerned activity. Thus for our example, the subset would consist of the critical resource $R_2$ for sequencing commitment, and the resources $R_2$ and $R_3$ (as used by the activity $A_2$ and $A_3$ respectively) for APV commitment. The heuristic is a partial look-ahead. Let's analyze the heuristic in detail with the example shown in figure 4.9:

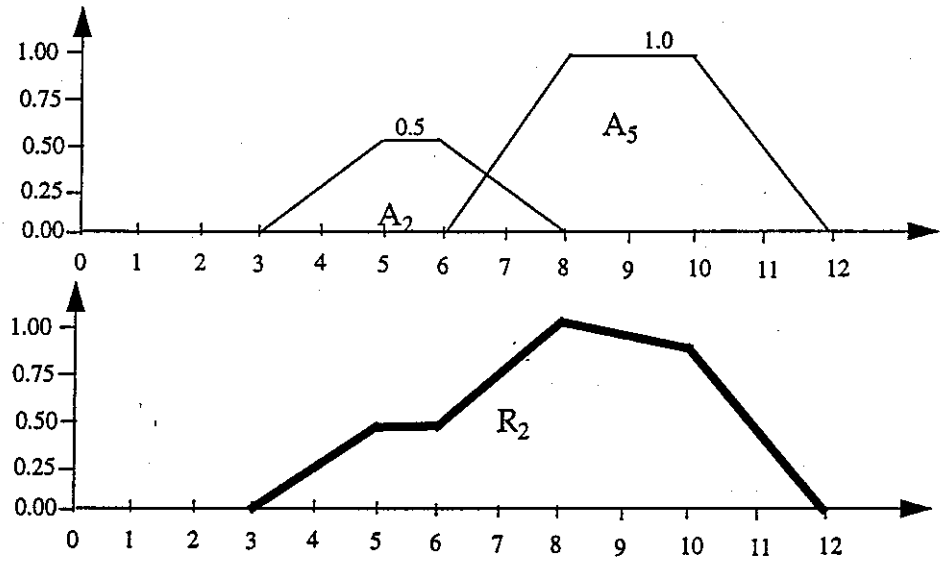
**Figure 4.9. Modified start and end times after the commitment $A_2$--> $A_5$**




- For the commitment $A_2$ --> $A_5$ (i.e., the activity $A_2$ must precede activity $A_5$) we first propagate the temporal values locally (only between these two activities). The effect of such propagation is shown in the following figure. The start time intervals for $A_2$ and $A_5$ have shrunk. The effect of such shrinkage would be reflected on the individual demand curves of both the activities as well as on the ADC of resources $R_2$ (the resource used by these two activities). Note that, this is local propagation and we are not propagating the effect of this commitment along the network. As shown in the figure 4.10, as a result of this commitment, the ADC of $R_2$ has moved up (although the area under the curve remains the same). Generate a new proba-

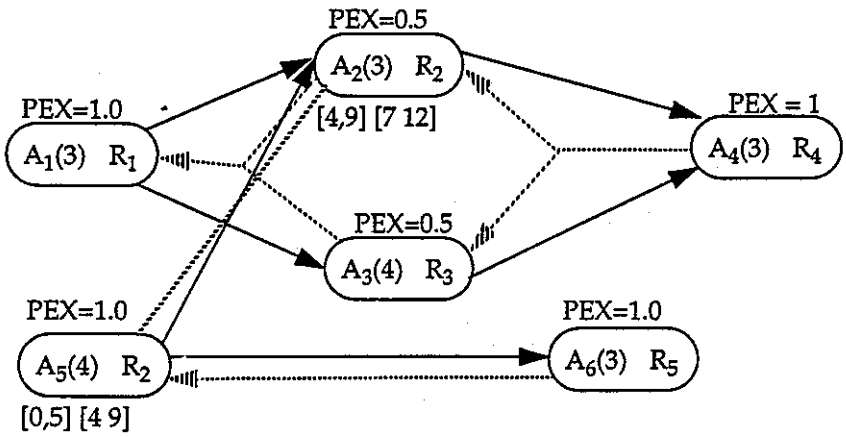
bility curve on the basis of this modified ADC and recalculate the PBMCC on resource $R_2$. Let this value be $P_{21}$.

**Figure 4.10. Modified Activity and Aggregate Resource curve after $A_2$ --> $A_5$**



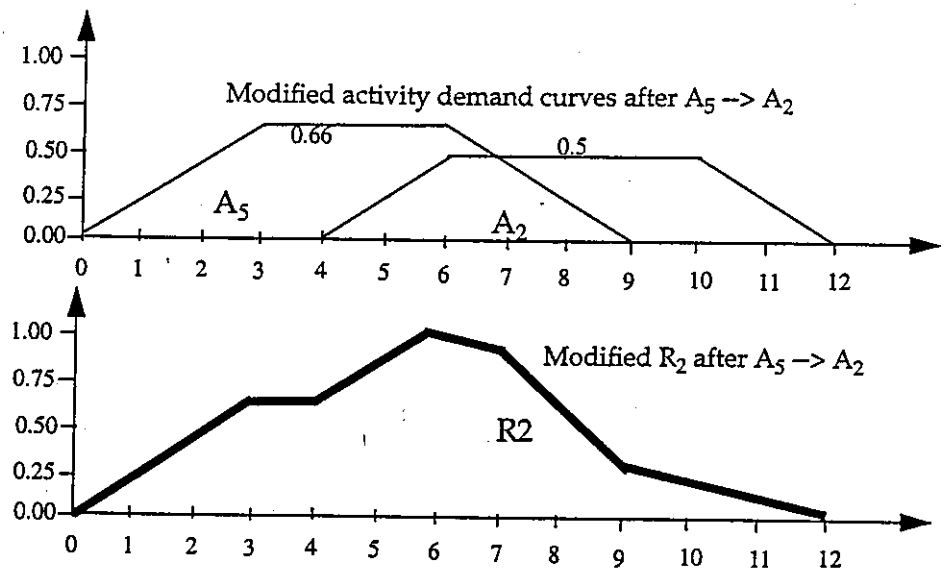

- The second commitment is $A_5$–>$A_2$, i.e. the activity $A_5$ must precede activity $A_2$. Like the earlier one we propagate locally (only to the activities $A_2$ and $A_5$) the effect of such commitment and update the intervals accordingly. The outcome of this commitment would be the shrinkage of the start time interval of the activities $A_2$ and $A_5$, as shown in the figure 4.11.

**Figure 4.11. Modified time intervals after the commitment $A_5$ --> $A_2$**



- The effect of such shrinkage would be reflected in the demand curve of the activity $A_2$ and $A_5$ as well as on the ADC of the resource $R_2$. Modify the ADC of $R_2$ to reflect the change and estimate the PBMCC for this modified curve. Let's say this is $P_{22}$. The modified curves are shown in the figure 4.12.

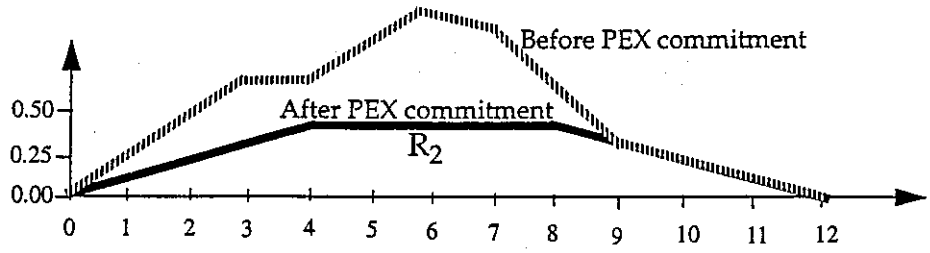
**Figure 4.12. Modified activity and resource demand curves after the commitment $A_5$ --> $A_2$**




The third commitment is to set the PEX value of $A_2$ equal to 0. The effect of this would not only be reflected on the aggregate demand curve of resource $R_2$ but also on the aggregate demand curve of resource $R_3$, as a result of the PEX propagation from $A_2$ to $A_3$. The effect on $R_2$ would be the reduction of contention at $T_2$ and the effect on resource $R_3$ would be an increase in contention because of the resulting increase in the demand for $R_3$ by the activity $A_3$. Thus, in order to get a measure of contention to compare with the first two commitment instances, we have to take into account the effect of this commitment on both the resource curves. For this we do the following.

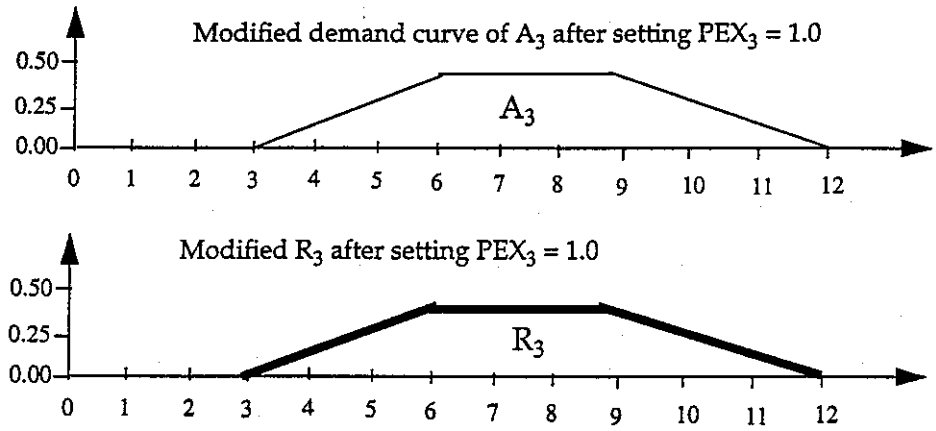
- Modify the ADC of resource $R_2$ after setting the $PEX_2 = 0$ (demand of the activity $A_2$ equal to 0) and estimate the probability of breakage of the maximum capacity constraint on this modified curve. Let this probability be $P_{23}$.

Figure 4.13. Modified resource demand curve of $R_2$ after setting the PEX of $A_2 = 0.0$




- Propagate the effect of $PEX_2 = 0.0$ to the other activities in the disjunct (i.e. to $A_3$). This would set the $PEX_3$ of the activity $A_3$ to be equal to 1.0. The effect of such commitment would be reflected on the individual demand curve of the activity $A_3$ as well as on the ADC of the resource $R_3$ (the curve would be raised). Modify the ADC of $R_3$ to reflect this change. Estimate the probability of breakage of the maximum constraint on the resource $R_3$. Let's say this is $P_{31}$. Take the maximum of $P_{23}$ and $P_{31}$. Let it be $P_{31}$.

Figure 4.14. Modified activity and resource demand curves after setting PEX of $A_3 = 1.0$




- Compare the values of $P_{21}$, $P_{22}$ and $P_{31}$. Pick up the commitment having the lowest value of $P_{ik}$. The logic behind choosing this is that the assertion of this commitment is going to reduce the contention most.

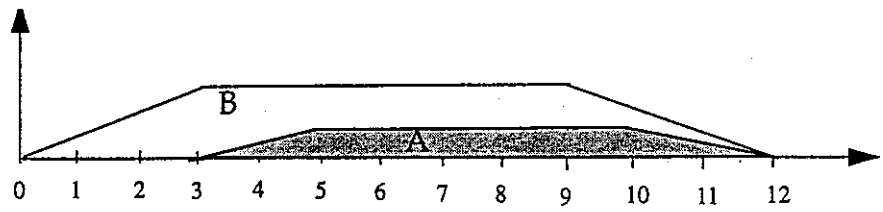
The heuristic proposed above involves recomputing the individual and aggregate demand curves, and the variance curve (for computing the probability of breakage of the maximum capacity constraint) for each possible alternative commitment instances. Note that, in order to get our desired probability values for the simple example stated above, we had to recom-

pute the aggregate demand curve for $R_2$ *thrice* and the aggregate demand curve of $R_3$ *once*. We also had to recompute the variance curves every time we recomputed any of the ADCs. Thus the heuristic, although cheaper than the first one, is still computationally expensive. This shows that, recomputing the PBMCC every time we modify our ADC although allows us to make an informed decision, but at a higher price. This motivated us to think of an alternative sets of heuristics that would give us a estimate of our decision without involving much of the overhead. The following section discusses two such heuristics.

## 4. 8 Texture Estimation based on Area

The intuition behind this texutre estimator is based on the observation that assertion of any of our commitment instance modifies either the shape or the actual area of the activity demand curves which in turn would modify the aggregate demand curves of the concerned resources. The changes in the respective area of the demand curves could give us a measure to compare the alternatives. However, there is one difficulty with this measure. The change in the area in not consistent for the two types of commitments (sequencing and PEX value). For APV (Assign PEX Value) commitment, the area under the demand curve changes, but for sequencing commitment the areas under the curve of the two activities remain the same. Thus although the proposed measure can be applied for APV commitment, we will not be able to utilize the change in the area information for comparing values in case of sequencing commitment.

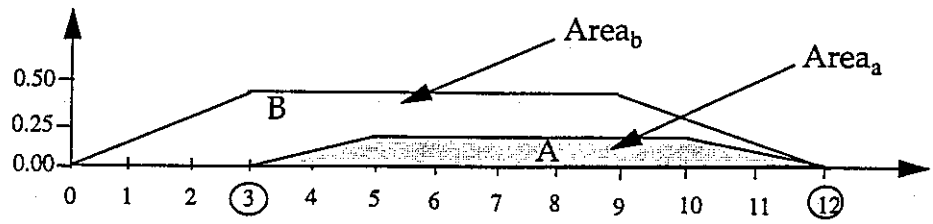
**Figure 4.15. Overlapped area between two activities**




This has prompted us to think about an alternative measures. Intuitively, one might propose to use the change in the overlapped area between the two activities as a prospective measure (see figure 4.15). The reasoning for this is as a result of the sequencing commitment there would definitely be a change in the overlapped area between the two activities . And we can utilize this change in the area to get an estimate of our decision. However, this heuristic would tend to select the commitment that reduces the overlapped area most. The result of this would be unnecessarily overconstraining the decision. To get around this problem, we

propose to have a different measure of the change in area. The measure is called 'area density', which is defined as follows:

**Figure 4.16. Area Density of activities A and B**




Area density of an activity i = $Area_i$ / ($lft_i$ - $est_i$)

Where

$let_i$ = latest end time of Activity i.
$est_i$ = earliest start time of Activity i
$Area_i$ = area under the demand curve of activity i

Thus the area density of activity A in the figure 4.16 is,

$$areaDen_A = Area_a/(12 - 3)$$

Let's see what this area density actually means. A simple interpretation would be, it is the height of the rectangle having the same area to that of the area under the demand curve, with a base spread over between the EST and LET. We can give another interpretation to this measure. Let us consider two activities A and B, having duration of $dur_A$ and $dur_B$ respectively. Let the EST and LET of the activities are $est_A$, $let_A$ and, $est_B$, $let_B$ respectively. Let the area of the demand curves of the two activities are $area_A$ and $area_B$ respectively. The slack for the ordering A-->B is given by:

$$slack_{AB} = let_B - est_A - dur_A - dur_B \qquad (1)$$

Area density of the demand curve of A corresponding to the ordering A-->B

$$\begin{aligned} den_A &= area_A/[let_A - est_A] \\ &= area_A/[(let_B - dur_B) - est_A] && [since, let_A = let_B - dur_B] \\ &= area_A/(let_B - est_A - dur_B) \\ &= area_A/(slack_{AB} + dur_A) && [From\ equation\ (1)] \end{aligned}$$

If the activity uses a single resource and the demand for the resource is a single unit then the area under the curve is equal to its duration.

Therefore, $area_A = dur_A$

Substituting this in the previous equation, we get,

$$den_A = dur_A/(slack_{AB}+dur_A)$$

By similar calculation we can show that area density of the activity demand curve B corresponding to the ordering A-->B,

$$den_B = dur_B/(slack_{BA}+dur_B)$$

Since the duration of an activity is a constant, we can say that, the measure 'area density' is proportional to the 'slack.' The above calculation shows that the proposed texutre estimator is another way of representing the concept of slack. But the advantage of this estimator is, where the slack heuristic fails to compare three alternatives mentioned earlier, this estimator allows us to do so. Since the slack heuristic has already showed strong performance on a set of benchmark problems, we are guaranteed to get at least the same level of performance for the type of problems that we are dealing with namely the alternative scheduling problems. Let's discuss the algorithmic step involved in our heuristic using this area based texture estimator. We would continue to use the example introduced in the figure 4.7.

- First, measure the area density between the activities $A_2$ and $A_5$ before any commitment is made. Let the initial total area density,

  $TotalAreaDen = Area_2/(let_2 - est_2) + Area_5/(let_5 - est_5)$

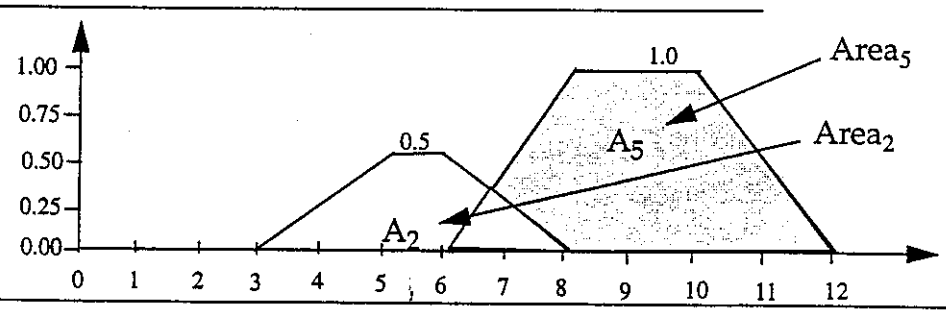
- Calculate the area density as a result of the commitment $A_2$ --> $A_5$ (figure 4.17). Let it be:

  *totalAreaDen1* = sum of the area densities of $A_2$ and $A_5$ after the ordering $A_2$--> $A_5$.

- Calculate the area density as a result of the commitment $A_5$ --> $A_2$. Let it be:

  *totalAreaDen2* = sum of the area densities of $A_2$ and $A_5$ after the ordering $A_5$--> $A_2$

**Figure 4.17. Modified activity demand curves of $A_2$ and $A_5$ after the commitment $A_2$ --> $A_5$**



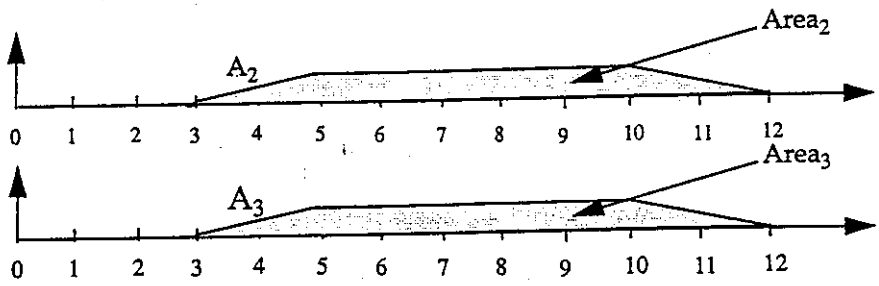
- Calculate the change in the area densities corresponding to these two commitments

$$changeDen1 = (TotalAreaDen - totalAreaDen1) / TotalAreaDen$$
$$= 1 - totalAreaDen_1 / TotalAreaDen$$
$$changeDen2 = (TotalAreaDen - totalAreaDen2) / TotalAreaDen$$
$$= 1 - totalAreaDen_2 / TotalAreaDen$$

- Calculate the area density corresponding to PEX Commitment. First measure the total area densities of the activities in the disjunctive node. For our case, this would be the area under the activity demand curves of $A_2$ and $A_3$. Let the value be,

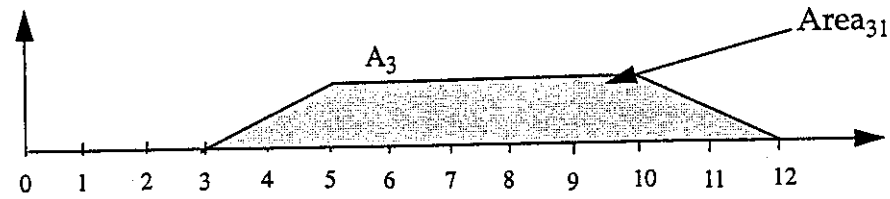
$$AltAreaDen = \text{sum of the area densities of all the alternative activities}$$
$$= Area_2 / (lft_2 - est_2) + Area_3 / (lft_3 - est_3)$$

**Figure 4.18. Original area density of the activities $A_2$ and $A_3$**




Set the PEX of $A_2 = 0.0$ and propagate the value to all its alternatives (in this case to $A_3$). The propagation is done to take into effect the changes in the area densities of the other alternative activities as a result of this commitment. Propagating PEX = 0 to other alternatives increases their PEX values, thereby increasing the contention on the resources used. For our example, the effect of propagating $PEX_2 = 0$ to $A_3$ is to increase its PEX value form 0.5 to 1.0.

**Figure 4.19. Modified area of $A_3$ after propagating PEX of $A_2 = 0.0$**




Now, calculate the sum of area densities of all the alternatives corresponding to the new PEX values,

*areaDenPex* = sum of the area densities of all the alternative activities after the PEX propagation (in our case this would be the area density of $A_3$ corresponding to PEX = 1.0)

- Calculate the change in the area density corresponding to the PEX commitment.

  *changeDenAlt* = (AltAreaDen - areaDenPex) / AltAreaDen
  = 1 - areaDenPex / AltAreaDen

- Select the commitment corresponding to the minimum of (*changeDen*$_1$, *changeDen*$_2$, *changeDenAlt*)

The advantage of this heuristic is that it uses only the activity demand curves instead of the resource demand curves. This significantly reduces the computational overhead associated with the decision.

It is evident from the above discussion that the more accurate the heuristic is, the more computationally expensive it would be. Thus there is a trade-off between accuracy and computational complexity. Also, note that one of the shortcomings of this (area density based heuristid) and the rest of the heuristics is that the texture measurements for these hueristitcs are focused only on a subset of resources. But it may well be the case that as a result of our commitment another resource which is outside of this set become critical. Thus unlike the one described in section 4.7.1, these heuristics would fail to track down such cases.

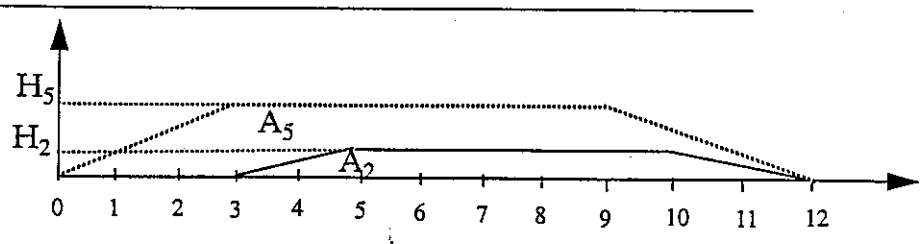
## 4.9 Texture Measurement based on Height

This heuristic is based on the observation that, assertion of any of our commitment instance modifies the maximum height of the activities' individual demand curves. The texture estimator is based on the change of the maximum height corresponding to each alternative decision. The algorithmic steps for the heuristic decision based on this estimator are as follows:

- First measure the maximum height of the demand curves of activities $A_2$ and $A_5$ before any commitment is made (figure 4.20). Let the initial total height is,

  *oriTotalHeight* = $H_2$ + $H_5$

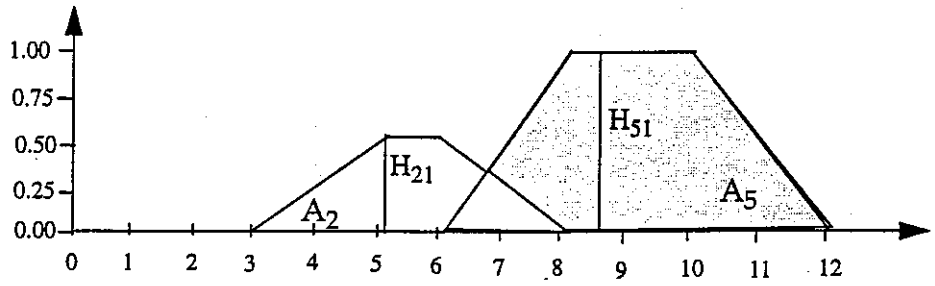
**Figure 4.20. Original maximum height of the activity demand curves of $A_5$ and $A_2$**



- Find the maximum height to these two curves corresponding to the commitment $A_2$ –> $A_5$ (figure 4.21). Sum these two heights. Let it be

```
totalHeight1 = sum of the maximum heights of A2 and A5
               after the ordering A2--> A5.
```

**Figure 4.21. Modified maximum height of the activity demand curves of $A_2$ and $A_5$**




- Calculate the maximum height of the demand curves of the activities $A_2$ and $A_5$ as a result of the commitment $A_5$ –> $A_2$. Let it be:

```
totalHeight2 = sum of the maximum heights the demand curves
               of A2 and A5 after the ordering A5--> A2
```

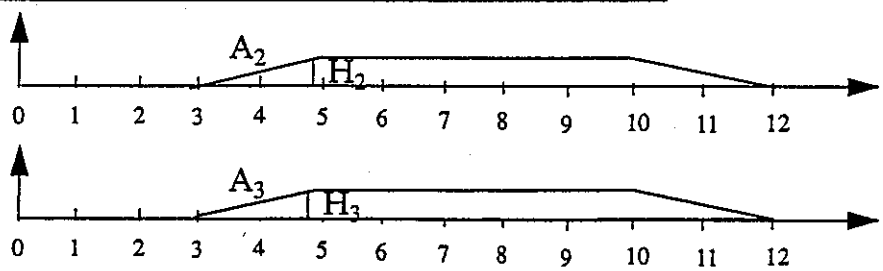
- Calculate the change in the maximum heights corresponding to these two commitments

```
changeHeight1 =(oriTotalHeight - totalHeight1) / oriTotalHeight
              = 1 - totalHeight1/oriTotalHeight

changeHeight2 = 1 - totalHeight2/oriTotalHeight
```

- Calculate the maximum height corresponding to PEX Commitment. First measure the total maximum heights of the activities in the disjunctive node. For our case, this would be the height of the activity demand curves of $A_2$ and $A_3$ (figure 4.22). Let the values be,

```
oriAltHeight = sum of the maximum heights of all the
                 alternative activities' demand curves
               = H2 + H3
```

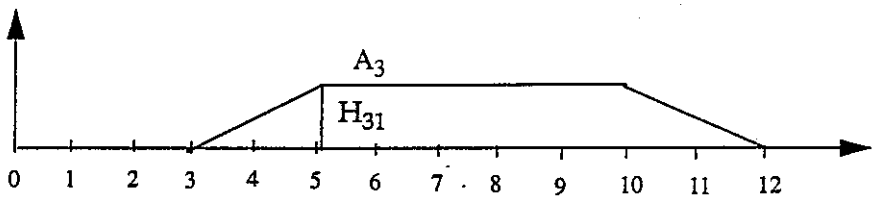
**Figure 4.22. Original maximum height of the activity demand curves of $A_2$ and $A_3$**



- Set the PEX of $A_2$ = 0.0 and propagate the value to all its alternatives (in this case to $A_3$). This would set the PEX of $A_3$ to 1.0 (figure 4.23). Now, calculate the sum of the maximum heights of all the alternatives corresponding to the new PEX values.

```
heightPex = sum of the maximum heights of all the alternative
            activities after the PEX propagation
```

**Figure 4.23. Modified maximum height of the activity demand curve of $A_3$ after propagating $PEX_2 = 0$**




- Calculate the change in the maximum height corresponding to the PEX commitment.

```
changeHeightAlt = (oriAltHeight - heightPex) / oriAltHeight
                = 1 - heightPex / oriAltHeight
```

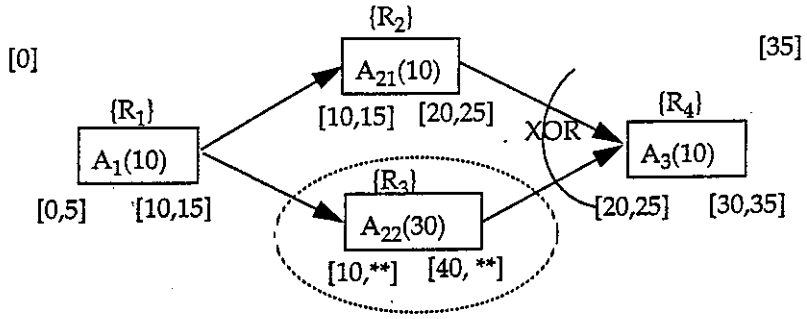
- Select the commitment corresponding to the minimum of (*changeHeight*$_1$, *changeHeight*$_2$, *changeHeightAlt*)

## 4. 10 Pruning of Alternatives During Propagation

For a job-shop scheduling problems with no alternatives, we are only interested in sequencing activities on the resources. But for the alternative scheduling problems, in addition to sequencing the activities on the resource we have to prune off the alternatives. Pruning of the alternatives are done during the search. We have already described how we pruned the alternatives in CBA algorithm (by deciding not to execute an activity in Case 3 of the modified CBA algorithm) and later with our new texture measurements (by assigning PEX = 0 to an activity). Alternatives are also pruned while achieving arc consistency by propagating temporal values across the network. This pruning is done if a temporal inconsistency is detected. For example, consider the following activity network with three activities (figure 4.24). One of the activity ($A_2$) has an alternative. During our initial forward propagation we set the EST of $A_3$ to 20 following the propagation algorithm of N-ary temporal constraints. But while propagating backwards, we notice an inconsistency. The EET value of activity $A_{22}$ is larger than the LST of activity $A_3$ which we are propagating to $A_{22}$ and $A_{23}$. Thus if we want to pro-

ceed with the propagation we cannot set the LET of $A_{22}$ to 25. The only alternative is to prunethe activity $A_{22}$. As $A_{22}$ has an alternative activity, we can set the PEX of activity $A_{22}$ to 0.0. Since we achieve our arc consistency by means of temporal propagation, we need to modify our temporal propagation algorithm for N-ary temporal constraints in order deal with such inconsistency. Our modified full backward propagation algorithm is as follows (we use same notations as listed in Chapter 3):

**Figure 4.24. Pruning of alternatives during temporal propagation**




```
/// first decide from which alternative activity to
/// propagate the LST value

if the number of activities in Act_to > 1
   for all activities in Act_to
   {
      if LST > LST_o
      LST_o = LST;
   }

for all activities in Act_from
{
    if(LST_o < EET)
       {
          set PEX = 0 and propagate to other alternatives;
          propagate the PEX value to dependents;
       }
       else
       {
             set LET = LST_o;
             propagate the value internally to the
                   StartTime variable.
             propagate the value to upstream activities;
       }
}
```

The above algorithm (full temporal propagation) is used is achieve arc-consistency before any commitments are made. After the assertion of a commitment we re-establish the arc consistency in the network with respect to temporal constraints by propagating new values

across the network. Pruning the alternatives is also done during this temporal propagation phase. Unlike the full temporal propagation algorithm, we need to modify both the forward and backward propagation algorithms for N-ary temporal constraints. The modified algorithms are as follows:

**Forward Propagation:**

```
/// first decide from which alternative activity to
/// propagate the EET value

 if the number of activities in Act_from > 1
        for all activities in Act_from
        {
           if EET < EET_o
           EET_o = EET;
        }

        if EET_o is not equal to the EET of Act_o
           RETURN;

    for all activities in Act_to
    {
      if(EETo < LST)
      {
        set PEX = 0 and propagate to other alternatives;
        propagate the PEX value to the dependent activities;
      }
      else
      {
        set EST = EET_o;
        propagate the value internally to the EndTime variable.
        propagate the value to downstream activities;
      }
    }
```

**Backward Propagation:**

```
/// first decide from which alternative activity to
/// propagate the LST value

if the number of activities in Act_to > 1
   for all activities in Act_to
   {
      if LST > LST_o
      LST_o = LST;
   }

for all activities in Act_from
{
  if(LSTo < EET)
  {
    set PEX = 0 and propagate to other alternatives;
```

```
        propagate the PEX value to the dependent activities;
    }
    else
    {
        set LET = LST_o;
         propagate the value internally to the StartTime variable.
        propagate the value to upstream activities;
    }
}
```

## 4. 11 New Commitment Type

We have already mentioned that, for alternative scheduling problems, in addition to the sequencing decision, we need to prune off the alternative choices as we move along the search process. We have also mentioned different stages where we prune off these alternatives. The pruning of alternatives is done by setting the PEX value of the activity to 0.0. Since assertion of this PEX value changes the search state, we can view it as a commitment. This calls for a need to have a new commitment type. For this, we introduced a new commitment called UEDCCommitment (Unary Existence Dependency Constraint Commitment). The effect of asserting this commitment is as follows:

- it sets the PEX value of the concerned activity to 0.0.
- propagates the effect of this value to all of its alternatives
- propagates the value to all of its dependent activities
- removes the activity from the network

The assertion of this commitment modifies the constraint network as we remove the activity and its dependents. The effect of such removal is propagated to other activities also. The way we do this is to reconstruct the activity network. Dynamically reconstructing the network is one of the complex issues that we had to deal with. The following section illustrates this point.

## 4. 12 Reconstructing the Constraint Network

The need for dynamically reconstructing the network arises in two cases. The first case is when we assert the UEDCCommitment and the second case is while we backtrack from this commitment. For UEDCCommitment, we need to set the PEX value of the concerned activity and propagate the value to its dependents in the network. Setting the PEX values to 0.0 means that these activities should not participate in the subsequent propagation process. In

other words, these activities will be considered to be non-existent to the other non-zero PEX activities in the network. This 'non-existence' of the activities in the network affects other activities which were connected to the removed activities by means of temporal constraints or were contending for the same resource. To take into account of these changes we have to update the start and end time variables of activities that were connected to these zero PEX activities. We also have to update the resource curves and the resource lists (lists that keeps tracks of all sequenced and unsequenced activities on a particular resource) that are associated with resource that the zero PEX activities use.

One way to make these activities non-existent is to delete them and reconstruct the entire network. But the problem associated with that is, during backtracking it becomes almost impossible to reinstate the deleted activities and their associated constraints. To get around this problem we use the PEX value of each activities in the network to decide whether or not the propagation will flow through that activity. A PEX value of 0.0 indicates that propagation need not flow through this as it is no longer a part of the active network. All of our propagation algorithm are modified to take this into account. While backtracking, we re-instate the PEX value to its previous non-zero value and thereby allowing any subsequent propagation to flow through the activity.

## 4.13 Infeasibility/Dead End Checks

The infeasibility/dead checks are done in the CBA algorithm (Case 3 of the modified CBA algorithm). As shown earlier, CBA algorithm was extended to incorporate the possibility of not executing an activity. But if none of the two activities being considered by CBA have any alternative, then the search reaches a dead end and we terminate the solution with infeasibility.

## 4.14 Termination Criterion

The termination criterion that we used for solving the alternative scheduling problems is, whether all the activities on each resource are sequenced or not. When all activities on every resource are sequenced, we terminate the search and returns the solution. Now it may so happen that, some of the alternative activities may remain in the final solution. If any such case arises, we keep the activity that has minimum duration and delete all other alternatives. Now, this would increase the PEX value to 1.0 for the activity whose alternatives are deleted. But it does not effect our solution, as our solution is still a valid one. Because, by sequencing

all the activities on the resource we are guaranteeing that none of the activities are executed simultaneously. So our resource constraints are not violated by the removal of the alternatives.

## 4. 15 Problem Declaration Language

ODO currently uses a problem description language called PODL (Odo's Problem Description Language). For detail description on PODL and its syntax, please refer to [Agrawal 96]. PODL allows a high level description of the problem which is then used as input to our problem solving module. In order us to solve the alternative scheduling problems, we need to represent these problems in PODL. But the current version of PODL cannot represent this type of problems. Various extensions were made in order for it to represent a wide variety of scheduling problems. Please refer to the **Appendix A** for detail description on these.

These extensions to PODL enable us to represent a wide variety of disjunctive scheduling problems. As we will see in the Chapter six, the above extensions can also be used to represent resource management problems. However, we will need one more extension to PODL for solving the resource management problem and this will be introduced later in the thesis.

## 4. 16 Summary

This chapter presents our work on extending and modifying the existing problem solving model of ODO for solving the alternative scheduling problems. We have identified nine areas where such extensions were made. Extension are done in algorithms, design and implementations. New heuristics are presented for solving the alternative scheduling problem. Extensions to PODL, the problem description language of ODO, for representing a wide variety of alternative scheduling problems are also presented.

# *Chapter 5 Experimental Results*

This chapter describes the results of our experiments with alternative scheduling problems. There is a dearth of benchmark problems for job-shop scheduling that uses alternative resources. In the absence of such benchmarks, we generated a suite of job-shop scheduling problems with alternative resources. The problem generation model was originally proposed by [Sadeh 91], and [Miyashita 94] later adopted it for his benchmark suites. We modified [Miyashita 94]'s problem to generate our benchmarks. The following sections describes these problems.

## 5. 1 Design of the Test Problems

The benchmark problem generated here are variations of the problem originally reported in [Sadeh 91] and used as a benchmark by a number of researchers (e.g., [Smith 93], [Muscettola 93]). However our problems are different from the original problem in one respect - our problem has substitutable resources for non-bottleneck resources. Following paragraph describes how these problems were generated.

We generated a set of 60 scheduling problems each comprising of 10 jobs. Each of these jobs are composed of 5 activities and 5 resources. All of these jobs follow a linear process plan which specified the temporal sequence in which the jobs are to be processed. Jobs within a single problem are not temporarily dependent of each other, i.e., there are no explicit temporal constraints between the jobs. However, they are constrained by the demand placed on the resources as these jobs share same sets of resources. Each job requires five resources. The sequence of activities vising the resources are randomly generated. Each job has one or two bottleneck resource which is visited after a fixed number of operations. The bottleneck resources and the alternative resource sets are predefined. Resources are randomly assigned to the activities. For activities that do not use the bottleneck resource have alternative resources to choose from. Thus if an activity is assigned a non-bottleneck resource $R_1$ and $R_1$ is a member of alternative resource set $\{R_1, R_4, R_7\}$, then the other two resources are also assigned to the activity as alternatives for $R_1$. No alternatives are specified for the activities that use bottleneck resource(s). This is intended to make the problem harder.

We used three different parameters to generate the problems. These are:

- Range parameter
- Bottleneck parameter
- Slack parameter

## 5. 1. 1 Range Parameter (RP)

The range parameter is used to generate the due dates and the release dates of each problem. The release date is the date when the job is released to the shop-floor and the due date is the date by which the job must be completed. The following uniform distribution is used to generate due dates:

$$(1+SP)(M)(U(1 - RP,1) \quad \text{(EQ 1)}$$

where,

U(a,b) = Uniform distribution between a and b;
SP = Slack Parameter;
RP = Range Parameter;
M = Makespan of the problem which is derived from the following distribution:

$$M = (n-1)\overline{dur}_{R_b} + \sum_{R=1}^{m} \overline{dur}_R \quad \text{(EQ 2)}$$

where,

$\overline{dur}_{Rb}$ = Average duration of activities using the bottleneck Resource $R_b$
$\overline{dur}_R$ = Average duration of activities using the Resource R
n = number of jobs
m = number of resources
$R_b$ = Bottleneck resource

The above estimate was first proposed by [Ow 85], and [Sadeh 91] later adopted it for his benchmark problems. Release dates are randomly generated by the following uniform distribution:

$$(1+SP)\ (M)(U(0,RP) \quad \text{(EQ 3)}$$

Three values of the range parameter are used to generate the problems. These are 0.0, 0.1 and 0.2.

### 5. 1. 2 Bottleneck Parameter (BK)

The bottleneck parameter, BK control the number of bottleneck resources. Two values of BK are used - 1 and 2. Half of the problems are generated with one bottleneck resource and the other half with two bottleneck resources.

### 5. 1. 3 Slack Parameter (SP)

The slack parameter was adjusted as a function of the range and bottleneck parameters to keep the demand for bottleneck resources closed to a hundred percent over the major part of each problem. Durations for activities in each order were also randomly generated. The slack parameter was empirically set as,

$$SP = 0.1 * (BK-1) + RP \quad \text{(EQ 4)}$$

The following table (table 5.1) shows the parameters setting for each class of the problem sets.

**Table 5.1. Characteristics of the six problem sets used in the experiment**

| Problem Set | Bottleneck Parameter | Range Parameter |
|---|---|---|
| Class-1 (r0/b1) | 1 | 0.0 |
| Class-2 (r0/b2) | 2 | 0.0 |
| Class-3 (r0/b1) | 1 | 0.1 |
| Class-4 (r0/b2) | 2 | 0.1 |
| Class-5 (r0/b1) | 1 | 0.2 |
| Class-6 (r0/b2) | 2 | 0.2 |

## 5. 2 Test Parameters

The following parameters were used for all the experiments reported in this chapter.

### 5. 2. 1 Critical Activity Selection

Our heuristic for identifying the most critical activity(ies) is called 'SumHeight'. The SumHeight heuristic identifies the resource having the highest resource contention at any particular time point. Unlike [Sadeh 91]'s approach, this heuristic does not calculate the contention over the average duration of the activities on the resource. Instead, it measures the heights of the aggregate resource curves at different time points in order to identify the highest demand point. The resource having the highest demand (maximum height) at a particular time point is identified as critical.

### 5. 2. 2 Commitment Selection

Our commitment selection heuristic comprises of two parts. First we try to decide the preferred sequencing of the two most contending activities by means of CBA algorithm. If CBA algorithm fails to give us a decision (i.e., if it says that neither sequencing is possible), then we use the 'area density' heuristic that was introduced in Chapter 4.

### 5. 2. 3 Backtracking and CPU Time

We used Limited Discrepancy Search (LDS) [Harvey 95] algorithm as our backtracking mechanism. LDS is particularly suitable for search processes that uses a very good heuristic which gives right decision most of the time but occasionally fails. Our maximum LDS discrepancy level was set to 1000 and the maximum limit of backtracks was set to 1000. The maximum CPU time was set to 600 seconds for all the test runs. If no solution is found between any of these limits of number of backtracks and CPU time (which ever is reached first), then the search is terminated with failure. The experiments were run on an HP 9000 machine running HPUX 9.05.

Following paragraph shows a sample PODL (problem description language for ODO) file describing the search parameters:

```
(commitment-manager : name LDS  :manager-class LDS
                    :heuristic-commit (SumHeight
                    :estimators (centroid area))
                    :backtracker (BTLDS
                    :num-backtracks 1000
                    :lds-max-discrepancy 1000)
                    :max-cpu-time 600
                    :termination sequence
(start scheduling)
```

## 5. 3 Experimental Results for Scheduling Problem with Alternative Resources

### 5. 3. 1 Combination of Area and Centroid Heuristic

The following table presents the results of our first set of experiments with the alternative resources having same duration. Ten problems from each of the six groups identified in table 5.1 were run using the test parameters described in section 5.2. The table 5.2 shows the results of these runs. Out of the sixty problems, we were able to solve 59 problems. One of the problems appeared to be infeasible. The experiments were run with a combination of area-density

and centroid based heuristics. Recall that, in the critical activity selection process (section 4.5)

**Table 5.2. Experimental results for the problems solved by the combined area density and centroid heuristic.**

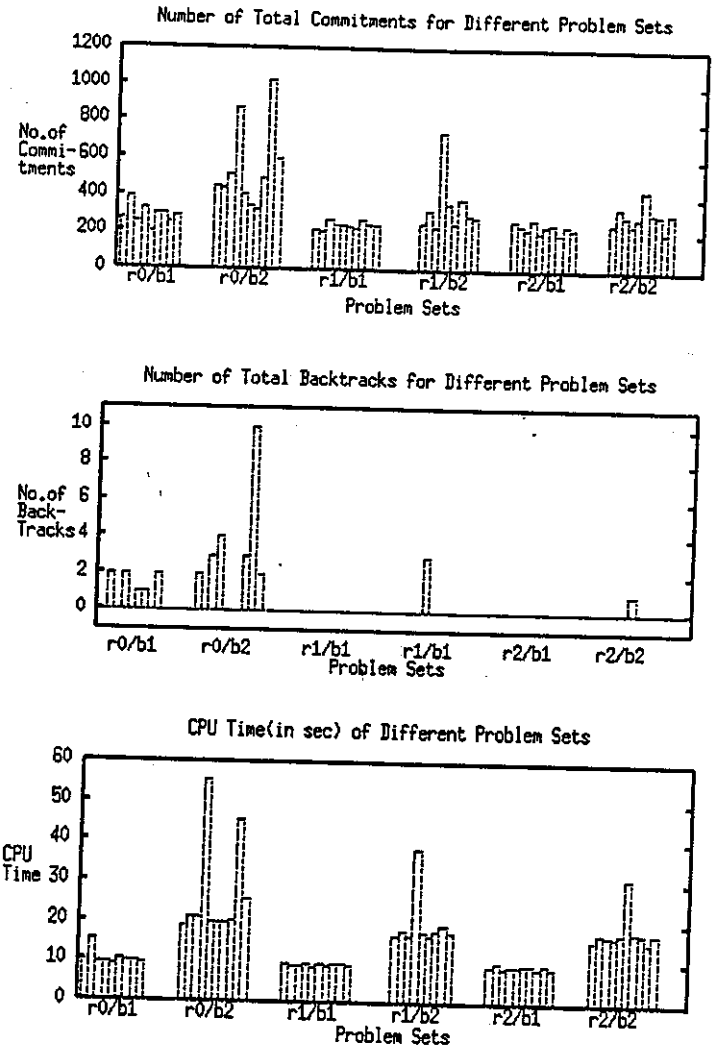
| No. of Probs. | Prob. Group | Range | No. of Bottle-necks | No. of Alter-natives | Make-span | Discre-pancy Level | Avg. no. of Commit-ments | Avg. No. of Backt-racks | Avg. CPU Time (Sec) |
|---|---|---|---|---|---|---|---|---|---|
| 10 | r0/b1 | 0.0 | 1 | 2 | 138 | 0 | 284 | 0.8 | 10.33 |
| 10 | r0/b2 | 0.0 | 2 | 3 | 154 | 0 | 546 | 2.4 | 26.69 |
| 10 | r1/b1 | 0.1 | 1 | 2 | 154 | 0 | 241 | 0 | 9.32 |
| 10 | r1/b2 | 0.1 | 2 | 3 | 171 | 0 | 352 | 0.3 | 19.93 |
| 10 | r2/b1 | 0.2 | 1 | 2 | 166 | 0 | 241 | 0 | 9.27 |
| 10 | r2/b2 | 0.2 | 2 | 2 | 182 | 0 | 302 | 0.1 | 18.49 |

if the heuristic was unable to find any alternative to the activities contending for the critical resource at the critical time point, the commitment selection process is left with two activities to make its heuristic commitment. But, if any of these two activities has an alternative resource/activity to choose from then the heuristic has to deal with three activities (the first two is the two most contending activities and the third one is the activity that has an alternative) for its heuristic commitment. The 'centroid heuristic' were used to make a sequencing commitment only if the heuristic were to deal with two activities. But if there were three activities involved in the commitment selection process the search algorithm used the 'area density heuristic' to make the next commitment.

Figure 5.1 shows the graphical representation of the results of our experiment. As shown in figure 5.1 and table 5.2, the average number of backtracks is less than 1 for five out of six problem groups. This is an indication that our texture based heuristics performed well in guiding the search in the right direction.

The number of commitments for two bottlenecks problems were comparatively higher than the one bottleneck problems. This can be explained by the fact that increasing the number of bottlenecks makes the problem more constrained and the heuristic spent longer amount of time in searching for the solution. CPU times for the two bottlenecks problems were comparatively higher than the one bottleneck problems. This is also attributed to the difficulty associated with these problem groups. However, the CPU time required by different problems are steady over certain problem groups. For example the average time spent for single bottleneck problems is close to 10 for the three different problem groups (r0/b1, r1/b1 and r2/b1).

Figure 5.1. Detailed results of the experiments for the combined area density & centroid heuristic



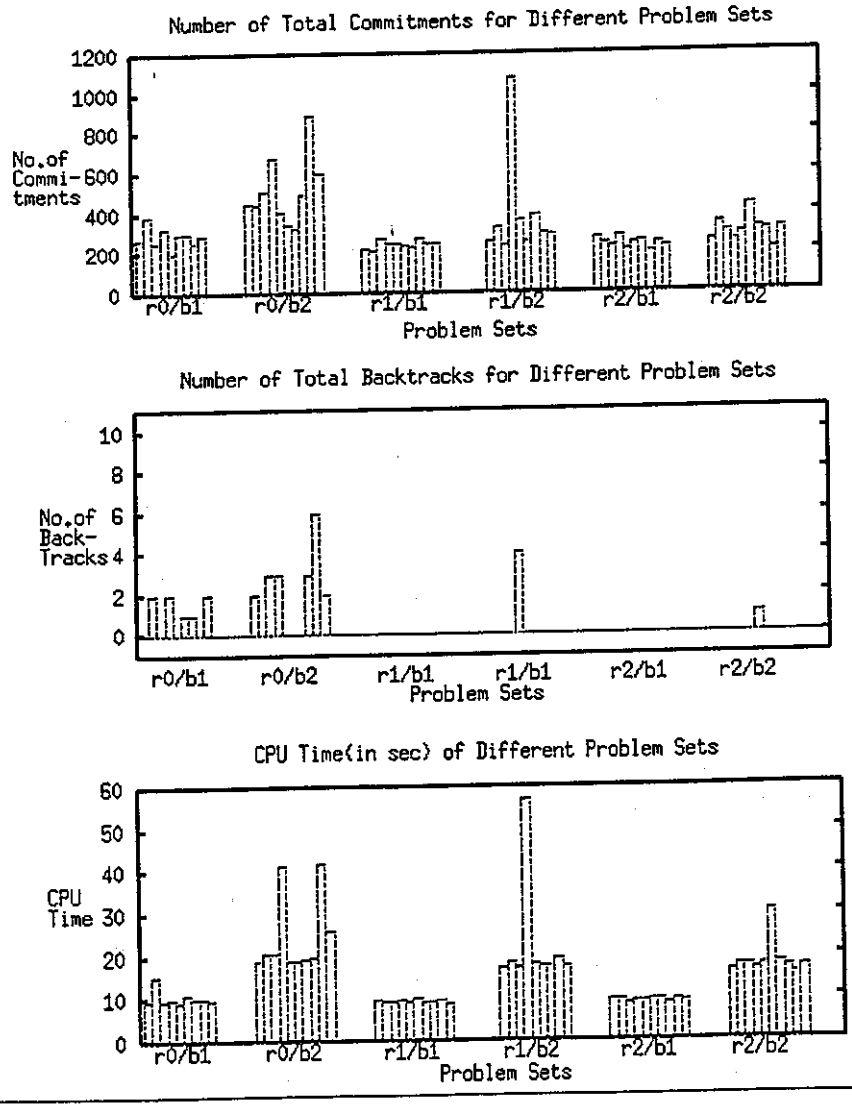

## 5. 3. 2 Area Density Heuristic

The next set of experiments were run with the same set of problems but only with the 'area density' heuristic. Unlike the previous run, we used the 'area density' heuristic even if our variable selection stage identified only two activities (i.e., no alternative exists among the most contending activities on the critical resource at the critical time point). Thus the commitment for sequencing decision is also based on this heuristic. Table 5.3 and figure 5.2 show the detail results of this experiment.

Table 5.3. Experimental results for the problems solved by area density heuristic

| No. of Probs. | Prob. Group | Range | No. of Bottle-necks | No. of Alter-natives | Make-span | Discre-pancy Level | Avg. no. of Commit-ments | Avg. No. of Backt-racks | Avg. CPU Time (Sec) |
|---|---|---|---|---|---|---|---|---|---|
| 10 | r0/b1 | 0.0 | 1 | 2 | 138 | 0 | 283 | 0.8 | 10.43 |
| 10 | r0/b2 | 0.0 | 2 | 3 | 154 | 0 | 513 | 1.9 | 26.61 |
| 10 | r1/b1 | 0.1 | 1 | 2 | 154 | 0 | 241 | 0 | 9.38 |
| 10 | r1/b2 | 0.1 | 2 | 3 | 171 | 0 | 388 | 0.4 | 22.17 |
| 10 | r2/b1 | 0.2 | 1 | 2 | 166 | 0 | 241 | 0 | 9.28 |
| 10 | r2/b2 | 0.2 | 2 | 2 | 182 | 0 | 301 | 0.1 | 18.43 |

Figure 5.2. Detail experimental results for the problems using the area density heuristic



Comparing the data in table 5.2 and table 5.3, it is evident that the results obtained from the two experiments using different heuristics are very similar. This is due to the fact that, both of these experiments used the area density heuristic for its commitment selection for the activities using non-bottleneck resources (as these activities have alternatives). The activities using the bottleneck resource have no alternatives. In the first experiment, the sequencing commit ment for these activities were made by using the centroid heuristic whereas in the second experiment the sequencing commitment were made using the area heuristic. Thus the variation of the results for these two experiments can be attributed primarily to the quality of the decision made by the two heuristics for the activities that are using bottleneck resources.

## 5. 4 Discussion

We don't have any benchmark results to compare the performance of our algorithm with others. However, our goal was to prove the validity of our approach towards solving the alternative scheduling problems. These experiments validates our approach by demonstrating that, our texture based heuristics can be used to solve scheduling problems with alternative resources. While analyzing the results of the experiments, we need to take into account two points. One is that, the introduction of alternative resources increases the solution space of the problems. The second point is attributed to our modelling approach for alternative resources. In our approach, we instantiate a new activity in the constraint graph for each alternative resource. Thus, although each of the test problems originally comprised of 50 activities, after the instantiation of the alternative activities, the constraint graph had 90 activities (for one bottleneck problem) and 110 activities (for two bottleneck problems). This tremendously increased the search space. So it would be unreasonable to compare our results with that of the original non-alternative scheduling problems generated by [Sadeh 91]. Despite this explosion of search spaces for problems with alternative resources, our results indicate that the CPU time required to solve the problems is reasonably small - an average of 9.68 sec for one bottleneck problem and 21.72 sec for two bottlenecks problems in the two experiments.

## 5. 5 Summary

This chapter presents the results of our experiments for a class of alternative scheduling problems namely the scheduling problems with alternative resources. We generated a set of benchmark problems for these experiments. At the beginning of the chapter we described the methodology for generating the test problems. Our problem generation approach is a varia-

tion of the approach used by [Sadeh 91] to generate his benchmark problem sets. A set of 60 problems were generated using our methodology. Two sets of experiments were run using these problems. In the first set, we used the combination of the area density and centroid heuristic, and in the second set we used only the area density heuristic for our heuristic commitment decisions. An analysis of search statistics are presented. The search parameters that were measured are number of commitments, CPU times and the number of backtracks. The results of these experiments validated our approach of using constraint heuristic search techniques in solving the alternative scheduling problems.

# *Chapter 6 Solving the Resource Management Problem*

This chapter defines the resource management problem, and outlines our approach to represent and solve this problem under the framework of constraint based problem solving. In the chapters 3 and 4, we showed the extension of ODO, both in the representation and in the problem solving methodology to solve the scheduling problems with alternative resources or alternative activities. The current chapter draws on those works and demonstrates how the earlier extensions can be applied to solve the resource management problem.

## 6. 1 Resource Management in Manufacturing

Consider a hypothetical manufacturing enterprise. It produces certain products. It receives orders on a periodic basis from its customers. Production is triggered by the arrival of an order. We can think of this as a 'make-to-order' type enterprise. When an order is placed, before a production activity is initiated, we need to get a quick answer to the question "given the amount of resource and the time we have, would it be possible for us to meet the order quantity and due dates?". In order to get the answer for this question, we need to do the following:

- **Derive the detailed materials requirement plan**

  The materials requirements plan gives us the detailed requirements, over time, of each assemblies, sub-assemblies, components and raw materials in order to satisfy the order. The MRP logic derives these requirements. It uses the product's BOM (Bill of Materials) and explodes it to determine the detail requirements.

- **Derive the purchasing plan**

  Purchase plan specifies what quantity needs to be purchased and when, in order to meet the requirements of the materials plan.

- **Derive the manufacturing order**

  Manufacturing order specifies the quantities that need to be manufactured/assembled in order to meet the requirements specified in the materials requirement plan. This includes specifying the time and the quantity of each production/assembly order.

- **Examine the feasibility of the manufacturing order**

  This steps involves checking the feasibility of the manufacturing order with respect to the resource capacity. This amounts to checking whether the specified quantities can be manufactured/assembled within the specified due date with the existing capacity at the shop floor.

- **Examine the feasibility of the purchase order.**

In traditional MRP systems, these steps are executed sequentially when, in fact, they should be considered concurrently. Let us illustrate this point. Let's consider that a MRP plan specifies that a quantity of 'part A' has to be produced in order to satisfy the order. It releases the order to the shop floor. During scheduling of the activities for the manufacture of part A, it is observed that we do not have enough capacity to schedule it. A MRP module would simply return the plan to be infeasible and ask for revisions. But the competitiveness in the market often prohibits the revision of such plan. Because, if the manufacturer asks for the extension of the due date, the customer might be inclined to choose another manufacturer for its products. Thus the goal of the manufacturer should be to satisfy the customer demand as much as possible by whatever means possible i.e., the manufacturer must explore all the possible alternatives. These alternatives might include, addition of new resources, sub-contracting to an outside supplier, or purchasing from an outside supplier. In many cases, these alternatives might be costlier. But, if the customers are willing to pay a higher price for the order, these alternatives are may be worth pursuing. The sequential approach of resource management fails to provide the valuable input about the alternative courses of action which might result in the satisfaction of customer demands. Our goal is to develop an approach to overcome this shortcoming. The following sections describes our approach.

## 6. 2 Problem Definition

The problem that we wish to solve is:

- the selection of alternative activities and
- the assignment of resource to these activities

with the objective of satisfying an order. In a manufacturing context, this translates into deciding whether to buy, manufacture/assemble, or draw from inventory the required amount of parts for satisfying an order. We model the inventory as a resource and defines the alternative of drawing inventory as an activity that consumes the inventory.

Given:

- a customer order specifying a product and quantity. Each order has an earliest release date, and a latest completion date. The earliest release date is the date before which the operations on the order/job cannot be started. This could be due to unavailability of raw materials or other physical resources, or due to some organizational policy. The latest completion date refers to the latest date after which the customer will not accept the order.
- the bill-of-materials of the product to be supplied
- a set of resources/machines on which production/assembly activities can be executed
- a set of process plans for producing the product. A process plan specifies a set of activities for producing a fixed amount of product, defines duration of the activities, temporal relationships among the activities, and the resource requirements of each activity. It also specifies all the alternative ways of satisfying an order, thus we have disjunctiveness in the process plan. Each operation can have both a conjunctive or disjunctive set of activities preceding or succeeding it. The start times of the activities in an activity network corresponding to a single part is constrained by the earliest release date and latest completion date of the corresponding customer order. The inventory activities are instantaneous, i.e. their duration is zero. No duration is specified for the purchasing activities as well. A fixed duration is assigned to all other activities.
- inventory on-hand for different parts, components and sub-assemblies.

The objective is to:

- identify a set of activities from the available alternatives that must be executed in order to satisfy the total demand of the customer order.
- assign resources to these activities such that all temporal and resource constraints are satisfied.
- assign quantities to the activities that will remain in the final solution so that the total demand quantities are met.
- if a purchasing activity is selected, assign the allowable time window for it by taking into account the order due date.

Once we have identified the activities that need to be executed, the remaining problem amounts to assigning a set of physical resources $R=\{R_1,....R_m\}$ and associated time or time windows to a set of activities $A=\{A_1,....A_n\}$ that belongs to a specific order $O_i$ for a part. The

problem is very similar to that of scheduling a set of jobs $J=\{j_1,....j_n\}$ (where each job $j_l$ consists of a set of activities $A_l=\{A_{11},.....A_{1n}\}$)on a set of physical resources $R=\{R_1,....R_m\}$.

The following three assumptions (other than those stated in the problem definition) were made:

- satisfying a demand from the existing inventory has higher priority to that of manufacture or assemble or purchase.
- manufacturing/assembling in house has higher priority than that of purchasing.
- we don't have a resource constraint on our purchasing activity.
- consumption of inventory is instantaneous, i.e., no duration associated with the inventory drawing activities.

Let us illustrate the problem with an example. Let us consider a product 'A' whose BOM tree can be represented as in figure 6.1. We will basis our discussion on the BOM tree. Figure 6.1 specifies that part A is assembled from two components B and C. The ratio 1:1 indicates that one unit of a particular component is needed to assemble one unit of the parent part. Given this BOM information, we can now build an activity network for the satisfaction of the demand for part A.

**Figure 6.1. Bill of Materials for Part A**

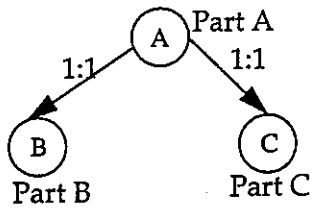



**Figure 6.2. A simple activity network corresponding to Part A**

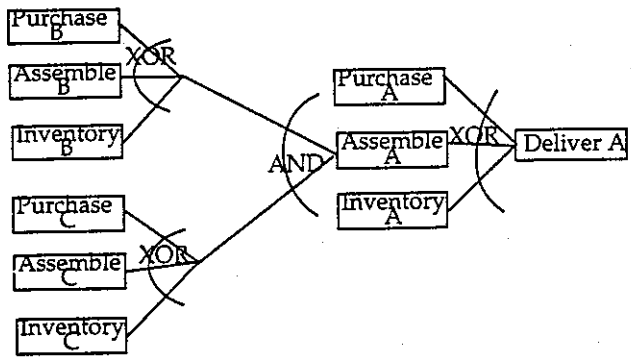


Figure 6.2 shows the corresponding network. The network uses some abstraction in its representation. For example, in figure 6.2, the assemble activities are represented by a single activity. Whereas, at detailed level of scheduling there could be series of activities that need to be executed for assembling a component, and we need to consider these activities while we are reasoning about the resource requirements. How detail this representation would be depends entirely upon the modeler. For example, it may not be worthwhile at this (material planning) level of decision making to include activities like set-up, tool change, coolant removal, fixture-removal etc., which exists at the shop floor level process plan. But one needs to include all activities that are critical from the resource management perspective. This is left entirely upon the discretion of the planner/modeler.

For our work, we are assuming that we know before hand what activities need to be included in the network for detail manufacturing/assembling activities. Note that these activity networks are not static. Rather, they change over course of solution process.

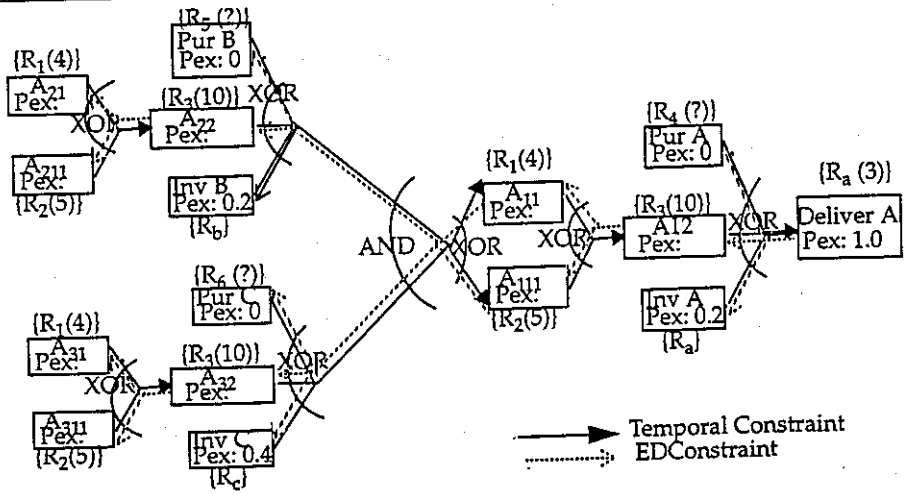
## 6. 3 Algorithm

Given an activity network shown earlier, our goal is to schedule the activities so that all temporal and resource constraints are satisfied and the demand for respective quantities are met. The resource management problem is very similar to the alternative scheduling problem that were introduced earlier in the Chapter 4 and the same methodology can be used for solving these two classes of problems. However, for resource management problem, PEX value has different connotation. Here we will use PEX values to indicate quantities assigned to the activity. Like the alternative scheduling problem, PEX value can assume a maximum value of 1.0 and a minimum value of 0.0. We represent the activity supplying the entire quantities of demand to be the activity having a PEX value of 1.0. All activities are assigned PEX values depending on their contribution to the maximum demand. Thus if an activity is assigned a PEX value of 0, it amounts to the non-existence of the activity in the network. The steps in solving the resource management problem are as follows:

### 6. 3. 1 Establish the constraint graph

The constraint graph is established from the given activity network. The activity network is a set of activities (based on the bill-of-materials) that contains all the possible ways to satisfy the requirement. The network contains conjunctive/disjunctive branches that specifies alternative ways of satisfying a demand for parts. These alternatives could be manufacture/assemble, purchase and draw from inventory. Let us expand the activity network shown earlier in figure 6.2. The resulting network is shown in the figure 6.3.

Figure 6.3. Expanded activity network for Part A




Note that in the figure 6.3 we have substituted each of the assemble activities of Part A, Part B and Part C with two activities. One of these two activities has an alternative activity. We have specified the resource requirements for each of these activities. Note that, no duration is associated with the purchasing activities. In fact the durations of the purchasing activities are one of the outputs from our solution process. Also note that, we have specified resources for the inventory activity. We view inventory as a resource consuming activity. Thus the activity 'Inv A' consumes resource $R_a$, which is the inventory of 'Part A'.

While instantiating the network, we assign PEX values to the activities corresponding to their available or assigned quantities. Thus the first activity in the network is assigned a minium and maximum PEX value of 1.0 (meaning it would supply the entire quantities mentioned in the order). All the inventory activities represent PEX values corresponding to the inventory on hand of the respective part. Thus if we have a demand for 100 units of product A and we have 30 units of inventory, than we assign a maximum PEX value of 0.3 to the activity 'Inv A', in our network. All other non-inventory activities in the network do not have any PEX value at this point.

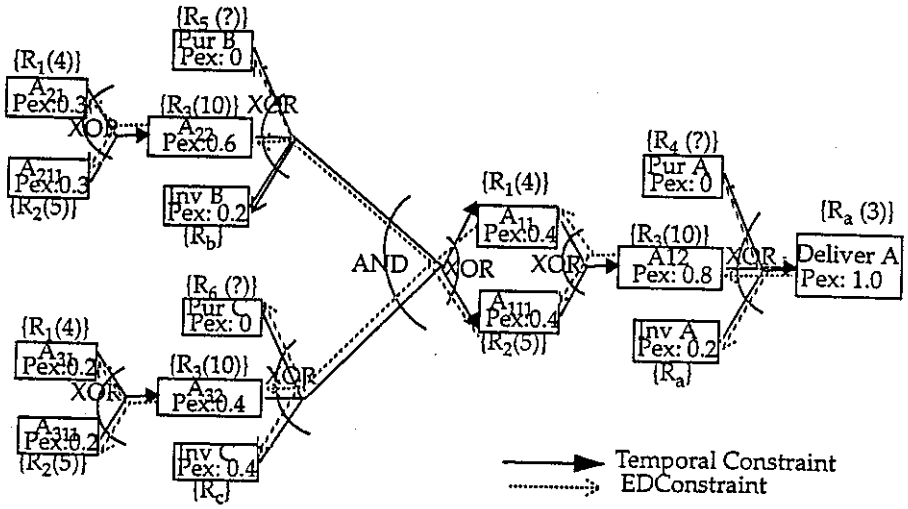
## 6. 3. 2 Demand Propagation

We then propagate the demand (PEX values) across the activity network. The activities with unassigned PEX values derive their value from the activities with known PEX values. For disjunctive activity we equally distribute the PEX value to all the alternatives. Note that, initially we would assign a minium and maximum PEX value of 0.0 to the purchasing activities.

This reflects our intention of not to purchase an item unless we have no other alternatives to pursue. However, for those products (such as raw materials) which have no other alternatives other than purchasing would be assigned a PEX value in proportion to their demand. Although the demand propagation is equivalent to the PEX propagation for alternative scheduling, the existence of 'purchase' and 'inventory' activities in the network complicates the propagation of these values. For this we modified the propagation algorithm for the n-ary EDConstraints.

At the demand propagation stage, we do the full PEX propagation. For full PEX propagation, we have to take care of the PEX values of the inventory activities. So before assigning PEX values to non-inventory activities, we have to subtract the inventory quantities (PEX of inventory activity) from the total demand. If it is found that the inventory on-hand is more than the total demand, then we prune off the corresponding portion of the activity network. For example, if in figure 6.3, the activity 'Inv C' has a PEX value higher than the required quantity of C (in this case the sum of PEX values of activities $A_{11}$ and $A_{111}$) then we prune of activity 'Pur C', activity $A_{31}$, $A_{311}$ and $A_{32}$. This is a manifestation of our assumption, that drawing from on-hand inventory is preferred over manufacturing or purchasing the item. Also, during our full temporal propagation, we would assign a PEX value of '0' to all purchasing activities reflecting our preference of not to assign any quantities to purchase activities unless we have no other alternatives. After the demand propagation (full PEX propagation) our activity network would have the following maximum PEX values:

**Figure 6.4. Activity network after PEX Propagation.**

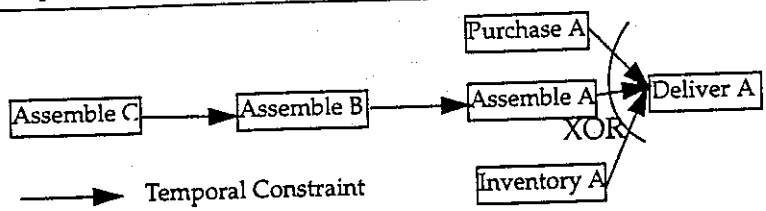


## 6. 3. 3 Temporal Propagation

The next step in solving the problem is to achieve arc consistency with respect to the temporal constraints. We have already introduced the temporal propagation algorithms for n-ary constraints in Chapter three. In order to take care of the fact that the duration of some of the activities in our network is unknown (i.e. purchase activities), we need to extend our full temporal propagation algorithms introduced in Chapter three. Extensions are made both in the forward and backward full temporal propagation algorithms. Now, while deciding about the EET or the LST of alternatives from which to propagate (refer to the section 3.4.2.1) we ignore the purchasing activities from such considerations. Also during our forward propagation to the downstream activities we set the EST and LST of the purchasing activities to be the same. And during our backward propagation we set the EET and LET of these activities to be the same. This is needed as we don't know apriori the duration of these activities. We then set the duration of the purchasing activity to be the difference between these two values (start time and the finish time).

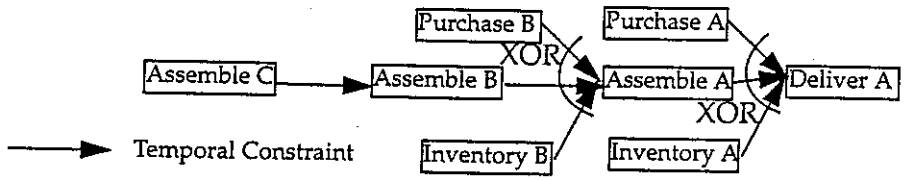
We also introduced an entity called 'root activity' in our propagation algorithm. The concept of root activity is used while turning off certain portion of the activity network in the event of non-feasibility of certain alternatives. Let us illustrate the definition of 'root activity'. Referring to figure 6.3, if we are unable to execute any of the activities $A_{21}$ and $A_{211}$ then we are unable to assemble the component B. What this means is, if we decide to assemble part B, then we should either execute activities $A_{22}$ and $A_{21}$ or $A_{22}$ or $A_{211}$. If because of due date constraint or resource unavailability we are unable to operate any activities from these sets ($\{A_{21}, A_{22}\}$ or $\{A_{211}, A_{22}\}$), then we have to look for alternatives other than assembly. Thus in the course of solution process if it is found that a downstream activity of $A_{22}$ and all its alternatives cannot be executed then we prune off the entire branch starting from $A_{22}$ and look for an alternative of $A_{22}$. The activity $A_{22}$ is defined as a root activity. All activities downstream of $A_{22}$ store this information. To further illustrate the concept of 'root activity', let us consider the activity network in figure 6.5:

**Figure 6.5. An example of a root activity**



In the figure 6.5, the root activity for both activities Assemble B and Assemble C is activity Assemble A. The activity 'Assemble A' has no root activity. But if the network is modified to be the following:

**Figure 6.6. The modified activity network consisting of a root activity**




Now the root of 'Activity C' is no longer 'Assemble A', it has now changed to activity 'Assemble B'. Also there is no root activity for 'Assemble B'. However, if we had an alternative resource for activity Assemble B, it would not have modified the root of 'Assemble C'. Thus root activity is associated to only those point in the activity network where we have an alternative which is not constrained by resource capacity and which does not contend for resource consumption with any of the manufacturing/assembling activities (e.g. purchasing or sub-contracting).

Extensions similar to *full* temporal propagation are also made to the forward and backward temporal (as opposed to full) propagation algorithms. As described in Chapter three, these algorithms are used to re-establish the arc consistency in the network after a commitment is made. No propagation is initiated from the purchasing activity. The time interval for such activity is only modified from the propagation coming from upstream activities (as purchasing has no downstream activities).

## 6. 3. 4 PEX Propagation

After achieving the initial temporal arc consistency we proceed to solve the problem as an alternative scheduling problem. As in the alternative scheduling problem we have two types of commitments - one for sequencing the contending activities on the critical resource, and the other one is to set the PEX of activity to 0. After each of these commitments we propagate their effects in the network. The algorithm for propagating the PEX values through the existence dependency constraints are introduced earlier in Chapter three. We need to extend the algorithm for propagating through the N-ary EDConstraint. The modified PEX propagation algorithm is much more complex than the one presented earlier in Chapter three. The com-

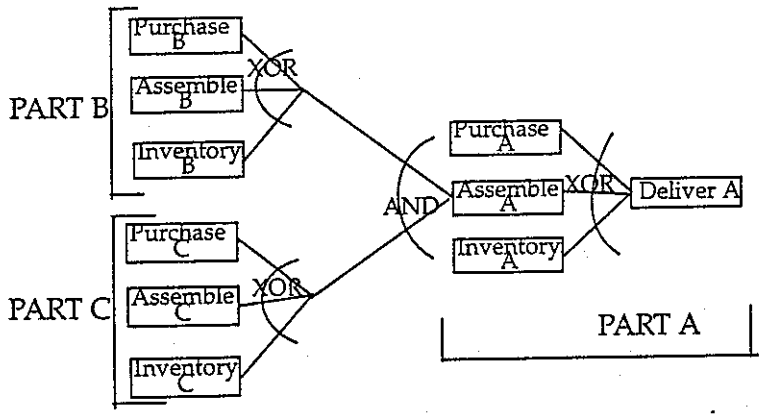
plexity is due to the fact that, we delayed the assignment of PEX values to the purchase activities. If during the solution process all alternatives except the purchase activity are pruned off, only then we assign the required PEX value to the purchase activities.

Note that, in order to solve the resource management problem we only need to extend the propagation algorithms (both temporal and PEX). Otherwise, we used the same representation framework and problem solving methodology as were used for solving the alternative scheduling problem. This is a manifestation of our algorithm's generality and its extendability to solve different classes of resource management problems.

## 6. 4 Generating the Problems

In order to prove the validity of our modeling and problem solving approach, we ran some experiments with our algorithms. To the best of our knowledge, there does not exist any benchmarks for resource management problems. So we generated our own problem sets. We used the problem generation methodology that was introduced in the Chapter five. We will use the product introduced in figure 6.1 as our required product whose demand has to be met.

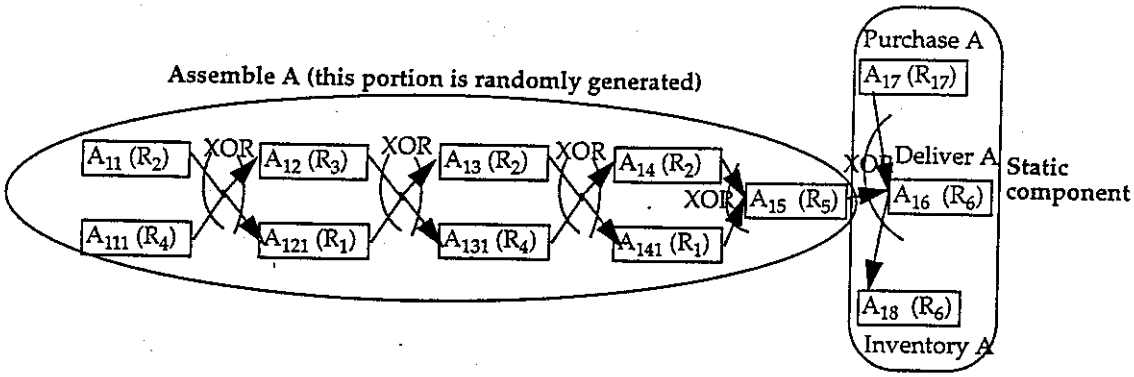
**Figure 6.7. The basic structure of the problem**




The activity network was generated based on the bill-of-materials of the product A and the available alternatives (both resource and activity). Figure 6.7 illustrates the basic structure of our problem sets. The figure shows at a level of abstraction the operations needed to manufacture/assemble Part A, Part B and Part C. As shown earlier in figure 6.1 Part A is assembled from Part B and Part C.

In our problem generation module we subdivided the activity network of figure 6.7 into two components. One is static, which is kept same for all problem instances. The other component is a variable and this was generated randomly using the problem generation methodology introduced earlier in Chapter five. The following figure explains the static and random components of our problem sets.

**Figure 6.8. Components of the Test Problem**




The static component comprises of purchase and inventory activities. For each of the assemble activities in the figure 6.7 (Assemble A, Assemble B and Assemble C) we randomly generated a series of five activities. These five activities use five resources. The temporal sequence of these activities are defined. The sequence of activities visiting the resources are randomly generated. Each of these five activities has one bottleneck resource which is visited after a fixed number of operations. For activities that do not used bottleneck resource have alternative resources to choose from. No alternatives are specified for the activities that use bottleneck resource. We randomly assigned resources to the activities. If the assigned resource is a non-bottleneck one, then an alternative resource is assigned to the activity from a predefined set. Thus if an activity is assigned a non-bottleneck resource $R_1$ and $R_1$ is a member of an alternative set $\{R_1, R_4\}$, then the another resource from the set (i.e. $R_4$) is assigned to the activity as its alternative. The alternative resources are then translated into alternative activities following the framework described in Chapter three. Figure 6.9 illustrates sample activity networks corresponding to the Part A, Part B and Part C that are used to construct the test problem.

We randomly assigned duration and resources to these activities. The bottleneck resource is predefined. Thus the bottleneck activities in all the three assembling activities use the same resource. Resources are also assigned to the inventory activities. The purchase activities in

the three branches of the network are assigned three different resources. This is done to reflect the fact that we are not associating any resource constraints on the purchasing activities (as we are not modeling the cost constraints). No duration is assigned to the purchasing and inventory activities. As mentioned earlier, inventory activities are zero duration activities and the duration of the purchasing activities are one of outputs from our solution. Each of the inventory activity in the network is randomly assigned a inventory quantity (PEX value). The final activity in the network, 'Deliver A' is assigned a PEX value of 1.0 (this means that this activity is supplying the entire quantities of the demand). These three branches of activity networks shown in figure 6.9 are then combined in the form shown in the figure 6.7 to construct a single problem. The resulting problem consists of 34 activities and 11 resources (including inventory and purchasing). We generated 10 problems based on these framework. We solved all of these problems. Rather than focussing on search statistics of the problems solved, we present different cases of a single problem instance and analyze the output given by our algorithm.

**Figure 6.9. A sample activity network for the Part A and Part B that are used to construct the experimental problems**

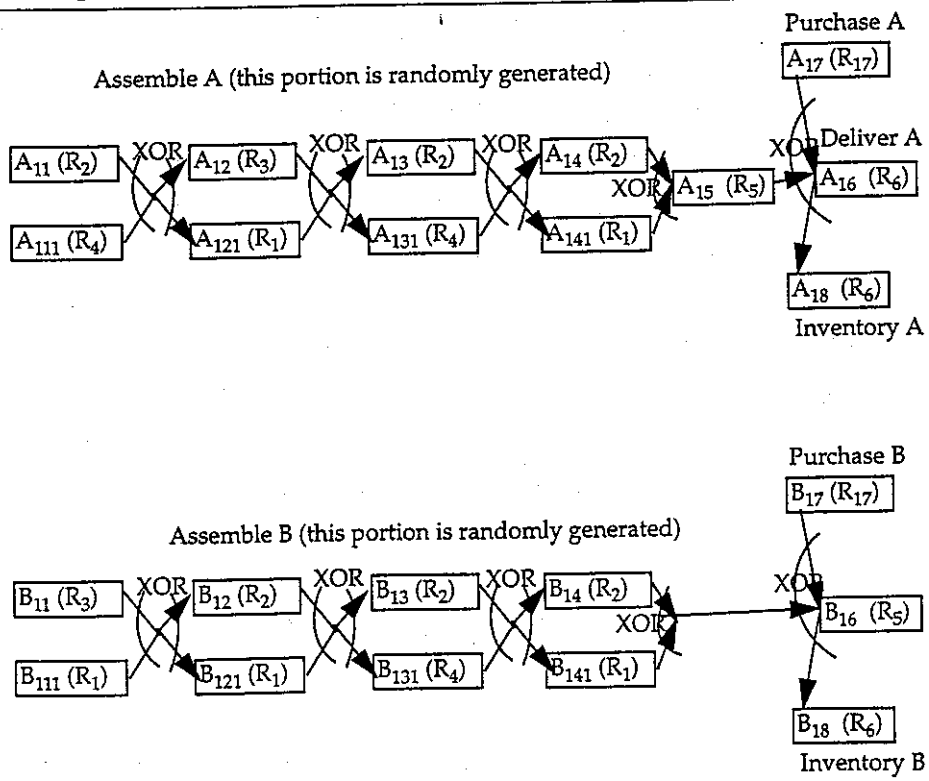
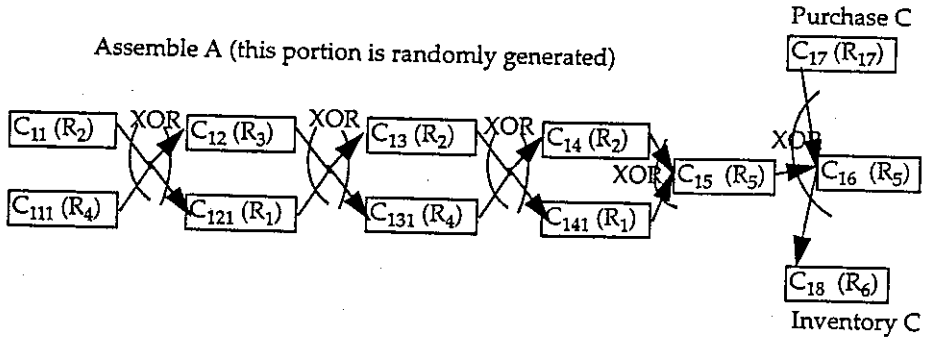


Figure 6.10. A sample activity networks for the Part C that is used to construct the experimental problems.




# 6. 5 Experimental Results

## 6. 5. 1 CASE I

Table 6.1, table 6.2 and table 6.3 give the parameters for the branch of the activity networks corresponding to the Part A, Part B and Part C (figure 6.9 and figure 6.10). Following tables use the same activity names as specified in the figures 6.9 and 6.10.

Table 6.1. **Parameter for the branch of the activity network corresponding to Part A**

| Activity | $A_{16}$ Deliver | $A_{17}$ Purchase | $A_{18}$ Inventory | *A* $A_{15}$ | *S* $A_{14}$ | *S* $A_{141}$ | *E* $A_{13}$ | *M* $A_{131}$ | *B* $A_{12}$ | *L* $A_{121}$ | *E* $A_{11}$ | *'A'* $A_{111}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Resource | $R_{16}$ | $R_{17}$ | $R_{16}$ | $R_5$ | $R_4$ | $R_2$ | $R_2$ | $R_4$ | $R_1$ | $R_3$ | $R_3$ | $R_1$ |
| Duration | 7 | ---- | ---- | 8 | 5 | 5 | 8 | 8 | 9 | 9 | 10 | 10 |
| PEX | 1.0 | ---- | 0.5 | --- | ---- | ---- | ---- | ---- | ---- | ---- | ---- | --- |

Table 6.2. **Parameter for the branch of the activity network corresponding to Part B**

| Activity | $A_{27}$ Purchase | $A_{28}$ Inventory | *A* $A_{25}$ | *S* $A_{24}$ | *S* $A_{241}$ | *E* $A_{23}$ | *M* $A_{231}$ | *B* $A_{22}$ | *L* $A_{221}$ | *E* $A_{21}$ | *'B'* $A_{211}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Resource | $R_{27}$ | $R_{26}$ | $R_5$ | $R_4$ | $R_2$ | $R_3$ | $R_1$ | $R_1$ | $R_3$ | $R_2$ | $R_4$ |
| Duration | ---- | ---- | 8 | 5 | 5 | 10 | 10 | 9 | 9 | 9 | 9 |
| PEX | ---- | 0.7 | --- | ---- | ---- | ---- | ---- | ---- | ---- | ---- | --- |

Table 6.3. **Parameter for the branch of the activity network corresponding to Part C**

| Activity | $A_{37}$ Purchase | $A_{38}$ Inventory | *A* $A_{35}$ | *S* $A_{34}$ | *S* $A_{341}$ | *E* $A_{33}$ | *M* $A_{331}$ | *B* $A_{32}$ | *L* $A_{321}$ | *E* $A_{31}$ | *'C'* $A_{311}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Resource | $R_{37}$ | $R_{36}$ | $R_{37}$ | $R_4$ | $R_2$ | $R_1$ | $R_3$ | $R_2$ | $R_4$ | $R_3$ | $R_1$ |
| Duration | ---- | ---- | 8 | 5 | 5 | 3 | 3 | 8 | 8 | 5 | 5 |
| PEX | ---- | 0.2 | --- | ---- | ---- | ---- | ---- | ---- | ---- | ---- | --- |

The problem was run with a scheduling horizon of [0, 87]. The following is the solution from our algorithm. Note that all inventory activities have been removed from the solution. We only used the inventory activities for our demand propagation:

**Table 6.4. Solution for the activity network corresponding to Part A of the problem.**

| Activity | $A_{16}$ Deliver | *A* $A_{15}$ | *S* $A_{14}$ | *S* $A_{141}$ | *E* $A_{13}$ | *M* $A_{131}$ | *B* $A_{12}$ | *L* $A_{121}$ | *E* $A_{11}$ | *'A'* $A_{111}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Resource | $R_{16}$ | $R_5$ | $R_4$ | $R_2$ | $R_2$ | $R_4$ | $R_1$ | $R_3$ | $R_3$ | $R_1$ |
| Duration | 7 | 8 | 5 | 5 | 8 | 8 | 9 | 9 | 10 | 10 |
| PEX | 1.0 | 0.5 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 |

**Table 6.5. Solution for the activity network corresponding to Part C of the problem.**

| Activity | *A* $A_{35}$ | *S* $A_{34}$ | *S* $A_{341}$ | *E* $A_{33}$ | *M* $A_{331}$ | *B* $A_{32}$ | *L* $A_{321}$ | *E* $A_{31}$ | *'C'* $A_{311}$ |
|---|---|---|---|---|---|---|---|---|---|
| Resource | $R_{37}$ | $R_4$ | $R_2$ | $R_1$ | $R_3$ | $R_2$ | $R_4$ | $R_3$ | $R_1$ |
| Duration | 8 | 5 | 5 | 3 | 3 | 8 | 8 | 5 | 5 |
| PEX | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 |

Let's analyze the solution. Referring to table 6.4 for Part A, we see that the purchasing activity does not exist in the solution. This means that the part can be manufactured/assembled in-house within the given due date. The demand for Part A was 1.0 (PEX of $A_{16}$) and we had 0.5 units on hand. So we have to assemble the rest. For this 0.25 units is distributed among the alternatives. Note that, within the given due date we have the option of choosing any of the available alternative resources. Thus all the alternatives remained in the final solution. Now, it is up to user/planner to decide which resource to use.

There are no activities in the final solution corresponding to Part B. This is due to the fact that Part B has an inventory of 0.7 whereas the demand for Part B is 0.5 (as 0.5 units of Part B is needed to assemble 0.5 units of Part A). So we don't need to execute any activity for this. For Part C (table 6.5), we also do not have the purchase activity in the final solution as the demand can be satisfied from in house assembly. Like Part B, the total demand for Part C for the current order is 0.5. As Part C already has an inventory of 0.2, we need to assemble an additional 0.3 units to satisfy the demand. This quantity is distributed among the alternatives. Like the Part A, we have the option of choosing any of the resources from the alternatives.

In addition to assigning quantities to different activities, the activities that needs to be executed are also sequenced on the respective resources. Thus the solution from our algorithm not only guarantees the satisfaction of demands within the given due date, it also guarantees the feasibility of the schedule.

## 6. 5. 2 CASE II

Let's now modify the parameters of the problem defined in tables 6.1, 6.2 and 6.3. Let's reduce the inventory quantity of Part B to 0.1. Since the demand form Part B is 0.5 units (as 0.5 units of B is needed to assemble 0.5 units of A), the effect of modifying the inventory quantity of B is that now we have to either assemble 0.4 units of B or purchase them from outside supplier. Following is the solution given by our algorithm:

**Table 6.6. Solution for the activity network corresponding to Part A of the problem.**

| Activity | $A_{16}$ Deliver | *A* $A_{15}$ | *S* $A_{14}$ | *S* $A_{141}$ | *E* $A_{13}$ | *M* $A_{131}$ | *B* $A_{12}$ | *L* $A_{121}$ | *E* $A_{11}$ | *'A'* $A_{111}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Resource | $R_{16}$ | $R_5$ | $R_4$ | $R_2$ | $R_2$ | $R_4$ | $R_1$ | $R_3$ | $R_3$ | $R_1$ |
| Duration | 7 | 8 | 5 | 5 | 8 | 8 | 9 | 9 | 10 | 10 |
| PEX | 1.0 | 0.5 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 |

**Table 6.7. Solution for the activity network corresponding to Part B of the problem.**

| Activity | $A_{27}$ Purchase |
|---|---|
| Resource | $R_{27}$ |
| Duration | [0 40] |
| PEX | 0.4 |

**Table 6.8. Solution for the activity network corresponding to Part C of the problem.**

| Activity | *A* $A_{35}$ | *S* $A_{34}$ | *S* $A_{341}$ | *E* $A_{33}$ | *M* $A_{331}$ | *B* $A_{32}$ | *L* $A_{321}$ | *E* $A_{31}$ | *'C'* $A_{311}$ |
|---|---|---|---|---|---|---|---|---|---|
| Resource | $R_{37}$ | $R_4$ | $R_2$ | $R_1$ | $R_3$ | $R_2$ | $R_4$ | $R_3$ | $R_1$ |
| Duration | 8 | 5 | 5 | 3 | 3 | 8 | 8 | 5 | 5 |
| PEX | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 |

The solutions for the Part A (table 6.6) and C (table 6.8) remain the same. For Part B (table 6.7), we have to purchase the required quantity as it is not possible to execute the assembly activities for Part B within the given due date. The duration assigned to the purchasing activity is an interval of [0 40]. This interval gives the flexibility of negotiating and choosing supplier from a set of alternative supplier for the Part B.

### 6. 5. 3 CASE III

Now let's increase the due date of the problem of case II. We arbitrarily increased the due date by 20%. Now the release date and the due date for the order is [0 100]. The resulting solution of the problem is now as follows:

Table 6.9. **Solution for the branch of the activity network corresponding to Part A**

| Activity | $A_{16}$ Deliver | *A* $A_{15}$ | *S* $A_{14}$ | *S* $A_{141}$ | *E* $A_{13}$ | *M* $A_{131}$ | *B* $A_{12}$ | *L* $A_{121}$ | *E* $A_{11}$ | *'A'* $A_{111}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Resource | $R_{16}$ | $R_5$ | $R_4$ | $R_2$ | $R_2$ | $R_4$ | $R_1$ | $R_3$ | $R_3$ | $R_1$ |
| Duration | 7 | 7 | 8 | 5 | 5 | 8 | 8 | 9 | 9 | 10 |
| PEX | 1.0 | 1.0 | 0.5 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 | 0.25 |

Table 6.10. **Solution for the branch of the activity network corresponding to Part B**

| Activity | *A* $A_{25}$ | *S* $A_{24}$ | *S* $A_{241}$ | *E* $A_{23}$ | *M* $A_{231}$ | *B* $A_{22}$ | *L* $A_{221}$ | *E* $A_{21}$ | *'B'* $A_{211}$ |
|---|---|---|---|---|---|---|---|---|---|
| Resource | $R_5$ | $R_4$ | $R_2$ | $R_3$ | $R_1$ | $R_1$ | $R_3$ | $R_2$ | $R_4$ |
| Duration | 8 | 5 | 5 | 10 | 10 | 9 | 9 | 9 | 9 |
| PEX | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 |

Table 6.11. **Solution for the branch of the activity network corresponding to Part C**

| Activity | *A* $A_{35}$ | *S* $A_{34}$ | *S* $A_{341}$ | *E* $A_{33}$ | *M* $A_{331}$ | *B* $A_{32}$ | *L* $A_{321}$ | *E* $A_{31}$ | *'C'* $A_{311}$ |
|---|---|---|---|---|---|---|---|---|---|
| Resource | $R_{37}$ | $R_4$ | $R_2$ | $R_1$ | $R_3$ | $R_2$ | $R_4$ | $R_3$ | $R_1$ |
| Duration | 8 | 5 | 5 | 3 | 3 | 8 | 8 | 5 | 5 |
| PEX | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 | 0.15 |

Note that the effect of increasing the due date is reflected on Part B (table 6.10). Now, instead of purchasing the Part B we are able to assemble the part in house and any one of the alternative resources can be chosen for the activities $A_{21}$, $A_{22}$ and $A_{23}$. The total quantity of 0.4 of Part B is equally distributed among the alternatives.

## 6. 6 Conclusion

This chapter delineates our approach to solve resource management problem. We have extended our problem solving framework of alternative scheduling problem to solve the resource management problems. Extensions are made both in the temporal and the PEX

propagation algorithms described earlier in Chapter three in order to use them for solving resource management problems. The extensions to these algorithms are discussed in the chapter. In order to demonstrate the feasibility of our approach we ran some experiments. As no benchmark problem exists for resource management problems, we provided a methodology to generate such problem sets. We generated a set of problems for a simple three component product. Each of these problems consists of 34 activities and 11 resources. The experiments with different variation of these problems are presented. The output of our algorithm is a set of activities and the quantity allocated to these activities in order to satisfy the demand for a given product within a given time frame. The activities are assigned quantities by taking into account the inventories on-hand and other resource and time constraints. Unlike, MRP system, our algorithm does not need a pre-defined lead time for purchasing activities. Instead the algorithm decides during the course of the problem solving whether we need to execute the purchasing activities or not and assigns a time interval to these activities.

# *Chapter 7 Concluding Remarks*

## 7. 1 Contributions

This project makes contributions in the following seven areas:

- Demonstrated the application of constrained based techniques for solving a broad class of resource management problems.
- Proposed and implemented a generic representation scheme for solving alternative scheduling problem that involve alternative resources, alternative activities and alternative process plans.
- Incorporated a new type of temporal constraints and the associated propagation algorithm to facilitate propagation through the alternative activities.
- Introduced the notion of probability of existence of an activity.
- Created propagation techniques to modify an activity's probability of existence by introducing a new type of constraint. These constraints express the dependency relationship of activities in a process plan.
- Developed heuristics techniques based on the use of texture measurements to take the probability of existence into account while choosing alternatives during the course of the search.
- Demonstrated a technique by which both the constructive and repair based search paradigm can be used in a single scheduling problem.

### 7. 1. 1 A Generic Framework for Alternative Scheduling Problem

Our modeling framework for solving alternative scheduling problems can be used across wide variety of alternative scheduling problems which include alternative resource with same durations, alternative resources with variable duration, alternative activities and alternative process plans. We have validated our approach to these classes of problems.

### 7. 1. 2 A New Type of Temporal Constraint

We introduced a new type of temporal constraints called N-ary constraints that can be used to bind 'm' number of activities with 'n' number of activities (where m and n >= 1). We presented the design of such constraints. The associated propagation algorithm for both forward

and backward propagations are presented. Various enhancements of these algorithm for tracking infeasibility, resource pruning and for solving resource management problems are also presented.

### 7. 1. 3 Probability of Existence (PEX)

We have introduced the notion of probability of existence (PEX) associated with each activity. A real variable is attached to each activity which indicates the probability that this activity would exist in the final solution. We latter extended the concept of PEX for resource management problem to represent the quantity allocated to each activity.

### 7. 1. 4 Propagation Techniques for PEX

We have also introduced the design of a new type of constraint for propagating the probability of existence of the activities. The associated propagation algorithm for these types of constraints are also specified. Various extensions of these algorithms for solving resource management problems are presented.

### 7. 1. 5 Heuristic Techniques for Alternative Scheduling Problem

We have developed new heuristic techniques for selecting the appropriate commitment while solving the alternative scheduling problems. We showed the failure of the existing heuristic measures to solve these problems and validated our approach on a set of alternative scheduling problems.

### 7. 1. 6 Solving Resource Management Problem

We extended the modeling and problem solving framework of alternative scheduling to solve a new class of problems that was not previously attempted by any researchers in the constraint satisfaction community. We provided the extensions to the propagation algorithms (both temporal and PEX) for solving this new class of problems. Our approach overcomes many of the shortcomings of the existing MRP systems which include infinite capacity assumption and fixed lead time. We demonstrated how the inventory management and scheduling problem can be modeled and solved under a unified framework. This integrated approach to solve these two classes of problems demonstrated the potential of applying CSP framework for different manufacturing management problems. The generic nature of our framework makes it suitable for applying the technique in planning and scheduling problems at different levels of decision making such as logistics planning, job-shop scheduling etc.

### 7. 1. 7 A Technique for Combining Two Research Paradigm

Two of the dominant research paradigm in constraint scheduling literature is constructive and repair based approach. The introduction of PEX variable to our activities, and incremental revision of these values along with our constructive approach to solve scheduling problems provided an example where two such paradigm can be combined in solving a single class of scheduling problem.

## 7. 2 Future Work

Although our approach has successfully demonstrated its applicability in solving wide variety of resource management problems, there are number of areas where future research can be pursued. These are:

### 7. 2. 1 Non-unit Capacity Resource

In order solve real life resource management problems, we need to extend the current work for resources having non-unit capacity. The interaction of PEX values and non-unit capacity resource would require extension to our existing heuristics.

### 7. 2. 2 Non-unit Resource Demand

Currently all our activities demand one unit of resource. Extending the framework for non-unit demanding resource would allow us to model and solve real life resource management problems.

### 7. 2. 3 Alternative Resource

Our framework for dealing with alternative resources involves instantiating new activities for each of the alternative resources. This rapidly increases the solution space. For example, for a single bottleneck 50 activity problem with 10 jobs (each having five activities), and having a single resource alternative for each non-bottleneck activities, the instantiation of alternative activities transforms it into 90 activities problem. Alternative ways of representing such problem is needed when we want to solve scheduling problems that comprises of activities having a larger domain of alternative resources.

### 7. 2. 4 Activity Lead Time

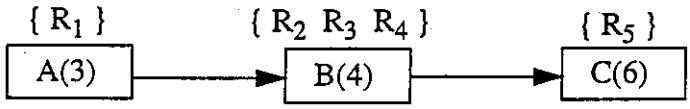
In our resource management problem, we only considered that the purchasing activities have unknown durations. All other activities in the network have known duration. This means that we associated a lead time to all manufacturing/assembling activities. Although elimi-

nating the fixed lead time assumption for purchasing activity significantly reduces the uncertainty associated with the problem, effort to solve the problem with unknown duration of the manufacturing/assembling activities would further increase the validity of the solution for uncertain environment.

# *Appendix A*

## A. 1 Alternative Resources with Same Duration

**Figure A.1. Alternative resource with same duration**



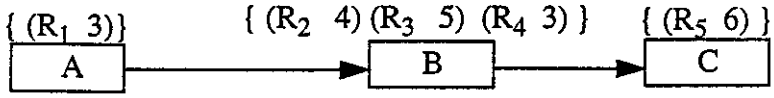

The corresponding PODL file is as follows:

```
(resource :name R1)
(resource :name R2)
(resource :name R3)
(resource :name R4)
(resource :name R5)

(activity :name A :duration 3 :uses R1 :depends-on B)
(activity :name B :duration 4 :uses-either (R2 R3 R4) :after A :depends-on
C)
(activity :name C :duration 6 :uses R5 :after B :pex 1)
```

The 'depends-on' specifies the dependency relationship. By default all activities' pex is set to 1.0. pex values other than 1.0 should be explicitly specified. The other activities derived their pex values through the propagation of the known pex values.

## A. 2 Alternative Resources with Variable Duration

**Figure A.2. Alternative resources with variable duration**




The problem can be represented in PODL as follows:

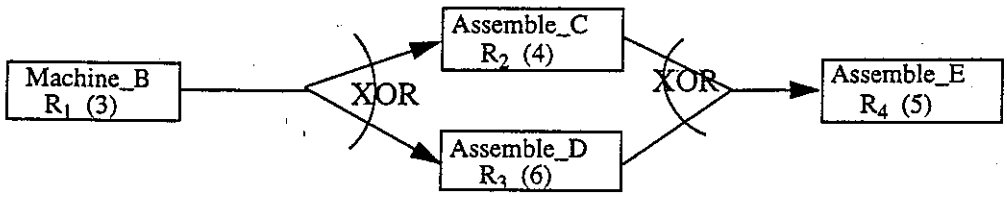
```
(resource :name R1)
(resource :name R2)
(resource :name R3)
(resource :name R4)
(resource :name R5)

(activity :name A :duration 3 :uses R1 :depends-on B)
```

```
(activity :name B :uses-either (R2 R3 R4) :dur-either(4 5 3) :after A
:depends-on C)
(activity :name C :duration 6 :uses R5 :after B :pex 1)
```

Note that, we have introduced two new terminologies, `uses-either` and `dur-either`. `uses-either` lists all the alternative resources and `dur-either` list the corresponding duration of the activity on the resources specified in `uses-either` list.

## A. 3 Alternative Activities

**Figure A.3. Alternative activities**



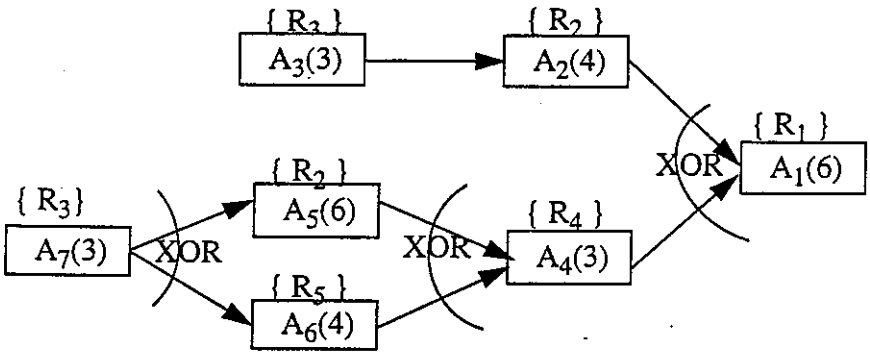

The corresponding PODL description is as follows:

```
(resource :name R1)
(resource :name R2)
(resource :name R3)
(resource :name R4)

(activity :name Machine_B :duration 3 :uses R1 :depends-on Assemble_C)
(activity :name Assemble_C :uses R2 :alt (Assemble_D) :duration 4 :after
Machine_B :depends-on Assemble_E)
(activity :name Assemble_D :uses R3 :duration 6)
(activity :name Assemble_E :duration 6 :uses R5 :after Assemble_C :pex 1)
```

Note that, in the definition of Activity_C we have introduced another new terminology, `alt`. `alt` lists the set of alternative activities corresponding to a single activity. Thus `alt` in the definition of Assemble_C states that one of the alternatives of this activity is activity Assemble_D. Alternately, we could have specified in the definition of activity Assemble_D that activity Assemble_C is its alternate activity. The same applies to the `depends-on` relationship. We specified in Assemble_C that it depends on Assemble_E. As the activity Assemble_D is the alternative of Assemble_C, it is also depended on the Assemble_E. We do not have to specify it again in the definition of Assemble_D. In the definition of Assemble_E, we specified that this activity is executed after activity Asssemble_C. Alternately we could have mentioned that Assemble_E is executed after Assemble_D.

## A. 4 Alternative Process Plan

Figure A.4. Alternative process plan




It might appear that we can combine the earlier extension to represent such alternative process plan problems. Thus the above problem can be represented as follows:

```
(resource :name R1)
(resource :name R2)
(resource :name R3)
(resource :name R4)
(resource :name R5)

(activity :name A7 :duration 3 :uses R3 :depends-on A5)
(activity :name A5 :duration 6 :uses R2 :alt (A6) :depends-on A4 :after A7)
(Activity :name A6 :duration 4 :uses R5)
(activity :name A4 :duration 3 :uses R4 :depends-on A1 :after A5)


(activity :name A3 :duration 3 :uses R3 :depends-on A2 :before A2)
(activity :name A2 :duration 4 :uses R2 :depends-on A1 :alt(A4) :before A1)
(activity :name A1 :duration 6 :uses R1 :pex 1)
```

But a closer look into the problem reveals the ambiguity associated with such representation. Note that, in the definition of activity $A_4$ we have specified that it should start after the activity $A_5$. In the definition of activity $A_5$ we have specified that activity $A_5$ has an alternative activity $A_6$. By stating the name of alternative we are implicitly saying that activity $A_4$ starts after both the activity $A_5$ and $A_6$. Now, in the definition of activity $A_3$, we specified that it ends before activity $A_2$ and the activity $A_2$ has an alternative, $A_4$. If we follows the same logic then, this would mean that, activity $A_3$ ends before the execution of activities $A_2$ and $A_4$. Which is obviously not the case. Thus a distinction has to made between the temporal relationship to all members and to a single member of the alternative set. Thus, in the above

example we have to differentiate the temporal relationship between activity $A_4$ and $A_5$, and its alternative $A_6$, and activity $A_3$ and activity $A_2$ (but not to its alternatives). This shows that, unlike the temporal relationship between two single activities, temporal relationship between alternative activities has cardinality associated with them. This observation is taken into account by associating cardinality to the temporal as well as dependency relationship. Following terminologies are introduced:

1. `after` - this indicates a temporal relationship of cardinality n-n.

2. `1_after` - this indicates a temporal relationship of cardinality 1-n. Where cardinality n is associated with the succeeding activities and cardinality 1 is associated with preceding activity.

3. `after_1` - this indicates a temporal relationship of cardinality n-1, were cardinality n is associated with the preceding activities and cardinality 1 is associated with the succeeding activity.

4. `before` - this indicates a temporal relationship of cardinality n-n.

5. `1_before` - this indicates a temporal relationship of cardinality 1-n. Where cardinality 1 is associated with the preceeding activity and cardinality n is associated with succeeding activities.

6. `after_1` - this indicates a temporal relationship of cardinality n-1, were cardinality 1 is associated with the preceding activity and cardinality n is associated with the succeeding activities.

7. `depends-on` - this indicates a dependency relationship of cardinality n-n.

8. `1_depends-on` - this indicates dependency relationship of cardinality 1-n. Where cardinality n is associated with the child activities and cardinality 1 is associated with parent activity.

9. `depends-on_1` - this indicates a dependency relationship of cardinality n-1, were cardinality 1 is associated with the parent activity and cardinality n is associated with the child activities.

With these new terminologies we will be able to overcome all ambiguities in the representation of problem. Thus the representation becomes:

```
(resource :name R1)
(resource :name R2)
(resource :name R3)
(resource :name R4)
(resource :name R5)

(activity :name A7 :duration 3 :uses R3 :depends-on A5)
(activity :name A5 :duration 6 :uses R2 :alt (A6) :depends-on A4 :after A7)
(Activity :name A6 :duration 4 :uses R5)
(activity :name A4 :duration 3 :uses R4 :depends-on A1 :after A5)

(activity :name A3 :duration 3 :uses R3 :depends-on_1 A2 :before_1 A2)
(activity :name A2 :duration 4 :uses R2 :depenes-on A1 :alt(A4) :before A1)
(activity :name A1 :duration 6 :uses R1 :pex 1)
```

Note the usage of new terminologies in the definition of activities $A_2$ and $A_3$. Alternately, we could have specified activities $A_2$ and $A_3$ as follows:

```
(activity :name A3 :duration 3 :uses R3 :depends-on_1 A2)
(activity :name A2 :duration 4 :uses R2 :depenes-on A1 :alt(A8 A4) :1_after
A3 :before A1)
```

# BIBLIOGRAPHY


[Adams 88] Adams, J., Balas, E. and Zawach, D., The Shifting Bottleneck Procedure for Job Shop Scheduling. *Management Science* 33 (3): 391-400, 1988.

[Aggarwal 74] Aggarwal, S.C. A Review of Current Inventory Theory and Its Applications. *International Journal of Production Engineering*. Vol.12, 1974. pp.443-482.

[Allen 83] Allen, J.F. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*. 26 (11):832-843, November, 1983.

[Azarmi 95] Azarmi N. and Smith R. Intelligent Scheduling and planning system for telecommunications resource management. *BT Technology Journal*, Vol 13. No. 1, January 1995.

[Baker 93] Baker, Kenneth R. Requirements Planning, *Handbooks in OR & MS*. Vol. 4, S.C. Graves et al., Eds., pp. 571-625. Elsevier Science Publishers B.V. 1993.

[Baptiste 94] Baptiste, Philippe. *Constraint Based Scheduling: Two Extensions*. MSc Thesis, University of Strathclyde, 1994.

[Baptiste 95] Baptiste, Philippe and Le Pape, Claude. A Theoretical and Experimental Comparison of Constraint Propagation Techniques for Disjunctive Scheduling. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, Montreal, Quebec, 1995.

[Baudin 90] Baudin, Michel. *Manufacturing Systems Analysis, With Application of Production Scheduling*. Yourdon Press Computing Series, 1990.

[Beck 94] Beck, J.C. A Schema for Constraint Relaxation with Instantiations for Partial Constraint Satisfaction and Schedule Optimization. *Technical Report TR-EIL-94-3, Enterprise Integration Laboratories, Department of Industrial Engineering, University of Toronto*, 1994.

[Beck 97] Beck, J. C., Davenport, A.J., Sitarski, E.M, Fox, M.S. Beyond Contention: Extending Texture-Based Scheduling Heuristics. *In Proceedings of AAAI-97*. AAAI Press, Menlo Park, California.

[Bensana 88] Bensana, E., Bel, G., Dubois, D. OPAL: A Multi-Knowledge-Based System for Industrial Job-Shop Scheduling, *Int. Journal of Production Research*. pp. 795-819, 1988.

[Blackstone 82] Blackstone, J.H., Phillips, D.T. and Hogg, U.L. A State-of-the-Art Survey of Dispatching Rules for Manufacturing Job Shop Operations. *International Journal of Production Research* 20(1):pp.27-45. January-February, 1982.

[Brown 94] Brown, D.E. Scherer, William T. (Eds). *Intelligent Scheduling Systems.* Kluwer Academic Publishers, 1995.

[Cheng 95] Cheng, Cheng-Chung. and Smith, Stephen F. Applying Constraint Satisfaction Techniques to Job Shop Scheduling, *Technical Report CMU-RI-TR-95-03, Robotics Institute, Carnegie Mellon University.* 1995.

[Chikan 90] Chikan, Attila(Editor). *Inventory Models,* Kluwer Academic Publications, 1990.

[Cleaves 96] Cleaves, Gerard W., March, Valdimir A. Strengthening weak links. *OR/MS Today,* April 1996, pp.32-37.

[Davis 94] Davis, E.D. ODO: A Constraint-Based Scheduler Founded on a Unified Problem Solving Model. *Technical Report TR-EIL-94-1, Enterprise Integration Laboratories, Department of Industrial Engineering, University of Toronto,* 1994.

[Dechter 88] Dechter, R. and Pearl, J., Network-Based Heuristics for Constraint Satisfaction Problems. *Artificial Intelligence* 34. pp.1-38, 1988.

[Dechter 89] Dechter, R. and Meiri, I., Experimental Evaluation of Preprocessing Techniques in Constraint Satisfaction Problems. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence.* 1989.

[Erchler 76] Erschler, J., Roubellat, F., Vernhes, J.P. Finding some Essential Characteristics of the Feasible Solutions for a Scheduling Problem. *Operations Research.* 774-483, 1976.

[Fox 83] Fox, M.S. Constraint-Directed Search: *A Case Study of Job-Shop Scheduling.* Ph.D. thesis, Department of Computer Science, Carnegie-Mellon University, 1983.

[Fox 86] Fox, M.S. Observations of the Role of Constraints in Problem Solving. *Proceedings of the Sixth Canadian Conference on Artificial Intelligence.* 1986.

[Fox 87] Fox, M.S. Constraint-Directed Search: *A Case Study of Job-Shop Scheduling.* Morgan Kaufman Publishers, Inc., 1987.

[Fox 89] Fox, M.S., Sadeh, N., and Baykan, C. Constrained Heuristic Search. *Proceedings of IJCAI-89*, 1989.

[Fox 90a] Fox, M.S. and Sadeh, N. Why Is Scheduling Difficult? A CSP Perspective. *Proceedings of the Ninth European Conference on Artificial Intelligence*, 1990.

[Fox 90b] Fox, M.S. Constraint-Guided Scheduling - A Short History of Research at CMU. *Computers in Industry.* 14 79-88, 1990.

[Fox 94] Fox, M.S. ISIS: A Retrospective. *Intelligent Scheduling*, Ed. Zewben Monte, Fox, M. S. Morgan Kaufman Publishers, Inc., 1994.

[Fry 92] Fry, T.D., J.F.Cox, and J.H.Blackstone, Jr., "An Analysis and Discussion of the Optimized Production Technology Software and Its Use", *Production and Operations Management*, 1, 229-242, 1992.

[Garey 79] Garey, M. R. and Johnson, D. *Computers and Intractability*, W.H. Freeman and Co., 1979.

[Goldratt 80] Goldratt, E.M. Optimized Production Timetable: *Beyond MRP: Something Better is finally Here.* October, 1980. Speech to APICS National Conference.

[Graves 81] Graves S.C. A review of production scheduling. *Operations Res.* Vol. 29, No.4, pp.646-675, Aug, 1981.

[Haralick 80] Haralick, R. and Elliott, G., Increasing Tree Search Efficiency for Constraint Satisfaction Problems, *Artificial Intelligence* 14, pp.263-313, 1980.

[Harvey 95] Harvey, W.D. and Ginsberg, M.L. Limited discrepancy search. In Proceedings of IJCAI-95. pp.607-613.

[Hillier 90] Hillier, F.S. and Lieberman, G.J., *Introduction to Operations Research.* Holden-Day Inc., San Francisco, CA, 1990.

[Johnston 89] Johnston, M. Knowledge-Based Telescope Scheduling. *Knowledge-Based Systems in Astronomy.* Springer-Verlag, 1989.

[Kumar 92] Kumar, V. Algorithms for Constraint Satisfaction Problems: A Survey. *AI Magazine* 13 (1): pp.32-44, 1992.

[Mackworth 77] Mackworth, A., Consistency in Networks of Relations. *Artificial Intelligence* 8, pp99-108, 1977.

[Mackworth 86] Mackworth, A., Constraint Satisfaction. *Encyclopedia of Artificial Intelligence*. Addison -Wesley, 1986, pp.205-211.

[Minton 92] Minton, S., Johnston, M.D., Phillips, A.B., Laird, P. Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*. 1990.

[Miyashita 94] Miyashita, Kazuo. *A Case-Based Approach to Improve Quality and Efficiency in Ill-Structured Optimization: An Application to Job Shop Scheduling*. Ph.D. Thesis, Faculty of Engineering, Osaka University.

[Muscettola 94] Muscettola N. Scheduling by Iterative Partitioning of Bottleneck Conflicts. *Proceedings of 9th IEEE Conference on AI Applications*, 1993.

[Nahamias 93] Nahamias, Steven. *Production and Operations Analysis*. Irwin, 1993. pp.184.

[Noori 95] Noori, Hamid and Radford Russel. *Production and Operations Management: Total Quality & Responsiveness*. McGrawHill Inc., 1995

[Nuijten 94] Nuijten, W. P. M. *Time are Resource Constrained Scheduling: A Constraint Satisfaction Approach*. Ph.D. Thesis, Eindhoven University of Technology, 1994.

[Panakwakar 77] Panakwakar S.S., and Iskandar, Wafik. A Survey of Scheduling Rules. *Operations Research* 25(1): pp.45-61, January-February 1997.

[Plossl 94] Plossl, George. *Orlicky's Material Requirements Planning*. Second Edition.

McGraw Hill Inc., 1994.

[Rodammer 88] Rodammer, Frederic A. and White, K. Preston. A Recent Survey of Production Scheduling", *IEEE Transactions on systems, man and cybernetics*, Vol.18, No.6, November/December 1988

[Sadeh 91] Sadeh, N., *Lookahead Techniques for Micro-Opportunistic Job Shop Scheduling*. Ph.D. thesis, Carnegie Mellon University, 1991. CMU-CS-91-102.

[Sadeh 94] Sadeh, N., Sycara, Katia, Xiong, Yalin. Backtracking techniques for the job shop scheduling constraint satisfaction problem, *Artificial Intelligence*, 76, pp.455-480, 1995.

[Smith 89] Smith, S.F., Ow, P.S., Matthys, D.C., and Potvin, J.Y. OPIS: An Opportunistic Factory Scheduling System. *Proceedings of International Symposium for Computer Scientists*. 1989.

[Smith 93] Smith, S., Cheng, Cheng-Chung. Slack Based Heuristics for Constraint Satisfaction Scheduling. *Proceedings of 11th National Conference on AI.* 1993.

[Tsang 93] Tsang, E. P. K. *Foundations of Constraint Satisfaction,* Academic Press, 1993.

[Tsang 95] Tsang, E. P. K, "Scheduling Techniques - a comparative survey", *BT Technology Journal,* Vol 13, No.1, January 1995, pp.16 - 28.

[Wright 74] Wright, Oliver, *Production and Inventory Management in the Computer Age,* Cahners Publishing, 1974.

[Zweben 94] Zweben, M. and Davis, E. and Daun, B. and Deale, M. Iterative Repair for Scheduling and Rescheduling. *IEEE Transactions on Systems, Man and Cybernetics.* 1994.