Building Robust Schedules using Temporal Protection - An Empirical Study of Constraint Based Scheduling Under Machine Failure Uncertainty

Hong Gao

TR - EIL - 95 - 2

Enterprise Integration Laboratory
Department of Industrial Engineering
4 Taddle Creek Road
University of Toronto
Toronto, Ontario, CANADA
M5S 1A4

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

Building Robust Schedules using Temporal Protection - An Empirical Study of Constraint Based Scheduling Under Machine Failure Uncertainty

by

Hong Gao

A thesis submitted in conformity with the requirements

for the degree of Master of Applied Science

Graduate Department of Industrial Engineering

University of Toronto

Abstract

Building Robust Schedules using Temporal Protection - An

Empirical Study of Constraint Based Scheduling Under

Machine Failure Uncertainty

Hong Gao

Master of Applied Science

Department of Industrial Engineering

University of Toronto

1996

We extend a Temporal Protection approach as an effort to deal with machine failure uncertainty in job shop scheduling. The approach focuses on predictively building robust schedules based on knowledge of uncertainty in advance. Activity duration and resource release time are protected so that the schedule generated is less likely to fail while being executed under the failure prone environment. The method provides the schedule dispatcher the flexibility to start the activity earlier than planned in reaction to schedule execution.

We implemented the method within a constraint-based scheduler ODO:TNG. Activity constraint graph and problem description language PODL were extended to describe the resource failure information. A constructive search mechanism is employed to build the predictive schedule. We developed a simulation mechanism to evaluate the schedule robustness under dynamic failures. Our experimental results showed that temporally protected schedules can be effectively executed under dynamic machine failures, incurring less total cost of work in process and job tardiness than if the schedule was unprotected. The amount of saving depends on the failure pattern and the criticality of the failure machine. This work serves as a basis for future research in the area of uncertainty management in constraint-based scheduling.

Acknowledgments

I am grateful to my supervisor, Professor Mark S. Fox, for providing guidance and enlightenment over the past two years. Also, many thanks to my second reader, Professor Mike Carter, for his useful comments.

Chris Beck, Sanket Agrawal, and Ioan Popescu were all team colleagues in the ODO:TNG project. The implementation can not be accomplished without their cooperation.

This research was supported, in part, by the Department of Industrial Engineering and the University of Toronto, the Manufacturing Research Corporation of Ontario, Natural Science and Engineering Research Council, Carnegie Group Inc., Digital Equipment Corp., Micro Electronics and Computer Research Corp., Quintus Corp., and Spar Aerospace.

I would like to thank my husband, Hongtao, for his years of support and encouragement.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

In a real-world environment, the probability of a precomputed schedule being implemented exactly as planned is quite low because of activity, resource, and execution uncertainties. Schedule disruptions are inevitable consequences. Various strategies have been researched. But no consensus has been reached as to the applicability and effectiveness of each strategy. Neither are there well structured benchmark data.

Given the reality that uncertainties are diversified and domain specific, many questions still need to be answered, such as, what are the uncertainties and should they be dealt with differently? which approach is more efficient, predictive scheduling, reactive scheduling, or combined? When does one approach out performs another? What is the relative effort and complexity associated with each approach? and etc.

The work in this thesis was motivated by the above observation. It extends a *Temporal Protection* approach which was first proposed by Chiang and Fox [Chiang 90]. The approach focuses on predictively building robust schedules based on advance knowledge of uncertainty. Activity duration and resource release time are protected so that the generated schedule is less likely to fail during schedule execution and incurs less execution cost. The method addresses machine failure uncertainty primarily. This work is based on the belief that predictive scheduling plays an important role with respect to resource failure uncertainty. However, it does not intend to minimize the need for reactive or iterative improvement scheduling. We focus on researching the predictive scheduling approach instead of comparing it with reactive results.

We implemented the technique as an extension to a constraint-based scheduler ODO:TNG. Problem description language PODL was extended to input machine uncertainty information, which are used in calculating temporal protection during the scheduling. A generic constructive search mechanism is employed to build the schedule. A constraint based simulation mechanism was developed to test the robustness of a generated schedule under dynamic failures. Initial experiments provided some insights into some of the relationships between uncertainty characteristics and effectiveness of protection. The thesis presents the problem description, explains the approach being researched, describes the implementation in ODO:TNG, demonstrates the experiment design on both dingle machine scheduling and job shop scheduling, and analyzes the experimental results achieved.

1.1 Job Shop Scheduling

Scheduling refers to a general class of problems where a set of activities (with due date assigned) need to be allocated on a set of resources over time while satisfying pertinent constraints. The domain of scheduling applications vary from manufacturing, transportation, public service to time tabling. *Job Shop Scheduling* is commonly encountered in the manufacturing domain, where resource usually refers to machine and activity refers to the production operation. Some assumptions include:

- no job can be processed on more than one machine simultaneously;
- no machine can process more than one job at a time;

- job data are known and deterministic;
- once a machine starts processing a job, the machine cannot process any other job until it finishes that job, and
- machines are available throughout the scheduling period.

Both operations research [Baker 92] and AI-based methods [Sadeh 91] [Zadeh 91] [Zweben 94] have been used in solving job shop scheduling problems in the past researches. Many heuristics have been designed to overcome the solving complexity that increases exponentially with the size of the problem. This difficulty increases even more as one or more of the assumptions are relaxed. For example, relaxing the third assumption leads to a stochastic job scheduling problem where job data is modeled as random variables. Relaxing the last assumption allows the machines to suffer random breakdowns. Both of the relaxations are modeled in this work. Detail problem representations will be discussed in chapters five and six.

1.2 Environment of Job Shop Scheduling

Manufacturing scheduling is subject to constant uncertainty in its environment. This section outlines some aspects of uncertainty that are inherent in the domain of scheduling. The needs for dealing with uncertainty in job shop scheduling are identified.

Job shop scheduling operates within a supply chain environment (Figure 1) which undergoes dynamic changes. The input to the chain is the production demand (order) from external or internal customers (other plants). Schedules are generated to accomplish the orders and are sent to the shop floor for execution. Materials that are either internally produced or bought from outside suppliers are consumed to produce products. Finished goods are stored in the warehouse until being transported to the customers who ordered them.

Job shop scheduling literally works at the lowest level of the production management. To obtain a clear view of the dynamics of the whole system, the next section will analyze the main uncertainty components of the supply chain environment. It is understood that trying to address them all would be too big a project for this thesis. Therefore, only those uncertainty factors that are associated with the job shop scheduling level will be identified for later focus.

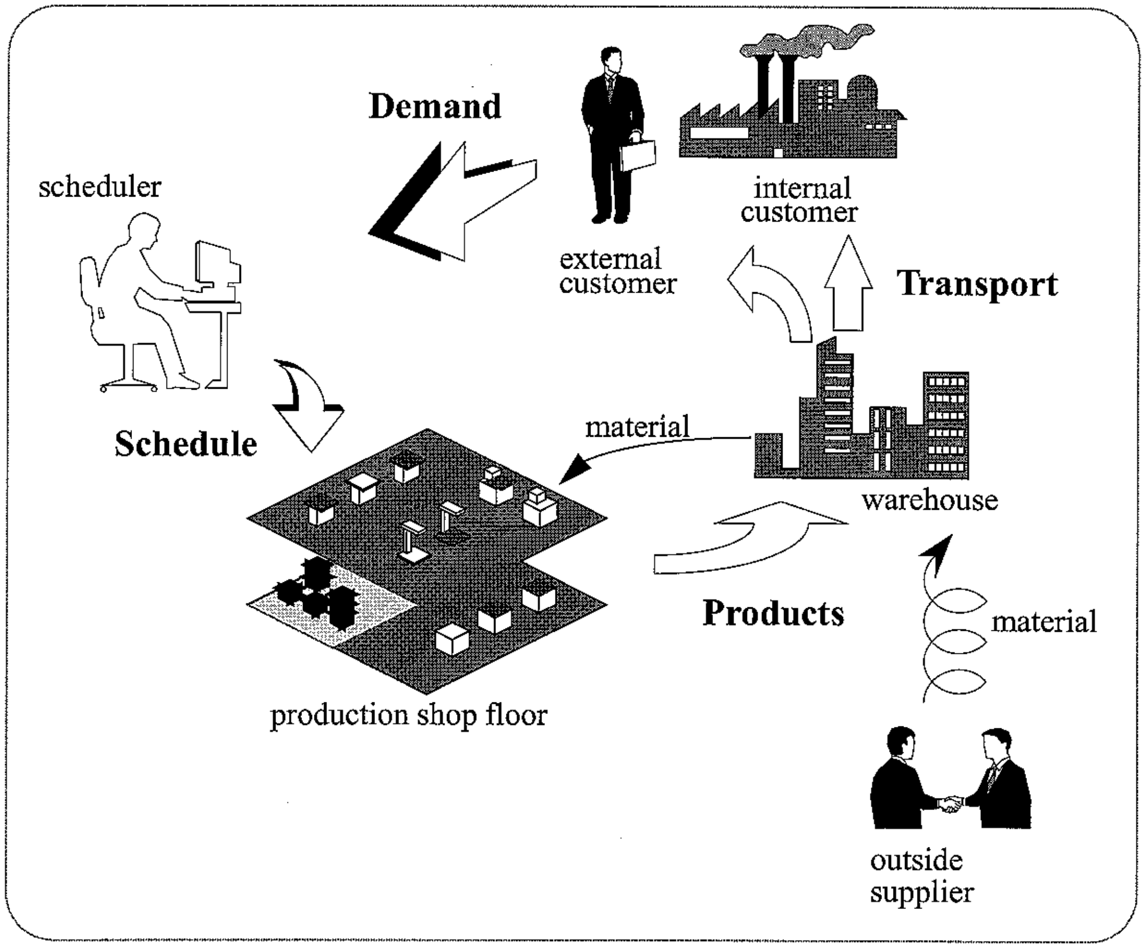


Figure 1. Environment of Scheduling

1.3 Scheduling Perspective of Uncertainty

This section tries to achieve an overall perspective of the uncertainty types within the scheduling domain, as well as its surrounding environment. The goal is not to endeavor solving all the problems, but to identify the diversity and understand the complexity of the problems before we go ahead and research on a certain aspect.

1.3.1 Levels of Uncertainty

Sources of uncertainty come from all aspects of the scheduling environment as in Figure 1. More specifically, they can be described to the following three managerial decision levels [Anthony 65] :

1. Strategic Level. (corporate high level decisions)

- Market forecast uncertainty [Rosenfield 87] ;
- Inadequate knowledge about competitors [Mamer 87] ;
- Customer acceptability of a new product [Holthausen 82] ;
- Decide investment and resource/labor assignment under uncertainties;
- Acquisition of new resource capacity under the uncertainty about the suppliers (quality, cost and deliver speed) [Hartl 92] [Paraskevopoulos 91] .

2. Tactical Level. (incomplete information at the plant level)

- Customer changes to orders (amount, due date, product specification, etc.);
- Task update by management - add/delete orders, etc.;
- Unreliable lead time of material supplier and quality of raw materials [Bassok 91] [Brennan 93] ;
- Inventory policy under uncertain supply and demands [Anupindi 93] [Ciarallo 94] [Tang 90] ;

3. Operational Level. (unsure or unknown information on the shop floor)

- Unsure of the processing time and setup/switching time, especially for new product which has never been manufactured before. [Drummond 94] [Wein 91] ;
- Machine/Tool breakdown [Wu 92] ;
- Material defects or not available at the time of request;
- Alternative routing [Hutchinson 94] [Masood 90] ;
- Defect/rework rate changes with the material quality and machine precision performance [Ciarallo 94] .

The higher level the decision is made, the bigger domain it will affect, and the bigger risk the manager is taking with respect to uncertainty in terms of investment and cost. Data is collected as much as possible at the corporate level so that the error in forecast can be minimized [Aykac 89] .

At the operational level, shorter decision cycle is required to respond to unexpected events. The following sections will discuss the demand and supply uncertainty first, which is widely researched at the tactical planning level. Operational level uncertainties will then be explored in more detail.

1.3.2 Uncertainty in Demand and Supply

Demand for the finished goods does not always conform as forecasted. It affects the production as well as the inventory. On the other hand, outside suppliers are not always delivering on time, or the delivered materials have quality defects and therefore can not be used for production or assembly. Interruption of the production might be caused. Machines and staffs may be left idling before satisfactory materials arrive. In reality, this problem is solved by diversifying suppliers and keeping safety stocks of materials in the inventory. When using the latter approach, there is the trade off between inventory holding cost and the cost incurred by not meeting production requirements. The objective of research in this area is to reduce the cost of inventory for both raw material and finished goods while meeting the demand. Higher level planning decisions is critical, such as better forecasting technique and flexible warehousing inventory control policy [Bassok 91] [Tang 90] . [Anupindi 93] suggests accommodating the supplier uncertainty by diversifying the supplier pool. [Brennan 93] simulated and evaluated demand and lead time uncertainties on material requirement planning systems. Introducing machine and process flexibility is also helpful for quick response to the demand variation [Hutchinson 94] . Note that it is difficult to forecast demand under some cases, such as a new product under trial production or a seasonal fashion good. If the change in the demand is so hugh that the available flexibility can not accommodate the change, rescheduling will be inevitable.

1.3.3 Uncertain Job Shop Floor

It is almost impossible to pre-plan all events on the shop floor in detail. Processing time may vary, materials may not be ready when they are required, machines and tools are subject to breakdowns and maintenance, alternative resource or routing might be available, and etc. The production process is thus uncertain and, possibly, unique for every production unit (part or batch of parts) going through.

The uncertainties that are inherent within a shop floor are introduced as following. These are general descriptions rather than formal definition.

1.3.3.1 Varying Activity Duration

In manufacturing reality, duration for an activity can only be approximated. The precision of approximation depends on the machine being used (manual operated or computer controlled) and the activity type. A production activity normally consists of several steps. Some typical ones are:

1. Prepare the materials and tools;
2. Load the part on the machine;
3. Processing;
4. Unload the part from the machine;
5. Clean up if necessary.

While the pure processing time for step 3 can be specified with relatively more accuracy, it is not true for other steps. Research experiences have reported [Wein 91]] [Hong 92] that activity durations can be modeled as random variables following certain distributions (uniform, normal, and etc.). For example, an activity duration which follows a uniform distribution of $U(m, \sigma)$, where m and σ are mean and variance of the random variable respectively, means that it is highly probable (depending on the confidence level) for the real duration to fall into the interval of $[m + \sigma, m - \sigma]$.

Under this circumstance, degree of uncertainty for the activity duration is the variance value of the distribution. For instance, in Figure 2, durations of activities 1 and 2 follow uniform distributions of $U(m_1, \sigma_1)$ and $U(m_2, \sigma_2)$ respectively ($\sigma_1 < \sigma_2$). Duration for activity 1 has a smaller deviation relative to its mean value and thus is described with more assurances.

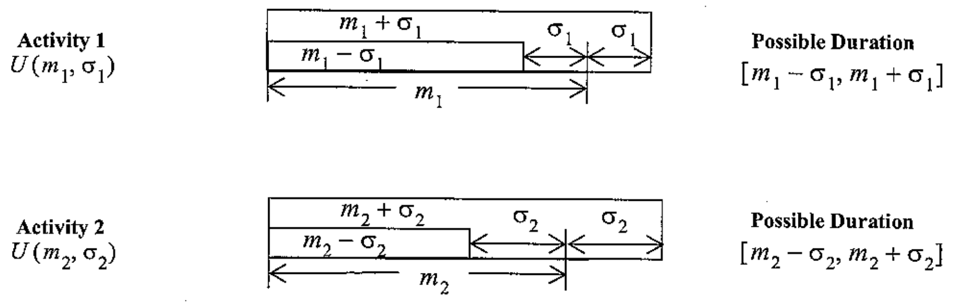


Figure 2. Activity duration modeled by uniform distribution

Note that we have included the setup time (steps 1 and 2) as part of the activity duration. The assumption is that setup time is relatively insignificant than processing time and can be included into the random model. If the setup time is sequence dependent and varies from activity to activity, they should generally be modelled as a separate activity and have their own random features. But it is true that sequence dependent setup time increases the dimensions of uncertainty and further complicates the scheduling environment.

1.3.3.2 Alternative Process Plan

A process plan depicts the sequence of activities that produce a product. Relevant information include: machines and tools required for each activity, the sequencing of those activities, and the materials being consumed and produced at each step. A process plan is uncertain if:

- Activities can be carried out by alternative resources, including machines, tools and materials, or:
- A set of activities can be executed in more than one sequence;

Basically, the more flexible a process plan is, the more alternative resources and routing it provides [Hutchinson 94] . This increased flexibility has both its benefits and challenges. On one hand, it allows quick response to schedule interruptions caused by unanticipated events such as machine breakdowns and defective materials/ tools. On the other hand, it increases the uncertainty dimensions in scheduling. This challenges the scheduling and control system for their ability to identify, plan, and then utilize alternatives when the need arises.

1.3.3.3 Unreliable Resource Performances

Resources that are on the shop floor can be classified into *consumable* ones and *reusable* ones (Figure 3). *Consumable* resources are assembled or transformed. For example, an assembly operation consumes some parts and produces a new product (another part or finished good). In another example, certain amount of cooling liquid are used for cooling down the machine during the production. *Reusable* resources include the machines, tools, fixtures and the staff who operate them.

The performance of resources are uncertain because *reusable* resource will fail from time to time and *consumable* resource are sometimes defective and cannot be used.

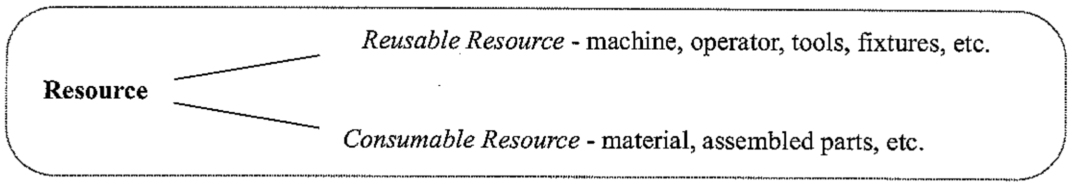

Resource
Reusable Resource - machine, operator, tools, fixtures, etc.
Consumable Resource - material, assembled parts, etc.

Figure 3. Reusable and Consumable Resources

- ***Machine Failure***

Machine breakdown is the most common type of uncertainty that interrupts the schedule execution. Quite often, the remaining schedule can not be carried out as planned. Orders can not be accomplished on time, resulting in customer dissatisfaction and financial lost. If the interruption is really serious, the impact could be carried on so that all pertinent orders are delayed as a result. For example, if failure delays the production of one part, assembly plant will not be able to start assembling as planned. The shipping date for the final product could be delayed.

Excluding intractable breakdowns that are caused by power off or unsuitable operations by inexperienced staffs, most machine failures can be patterned. As research in preventive maintenance revealed [Lewis 94] , machine failure characteristics in its life time can be represented as a tub-curve (Figure 4). Failure rate of the new machine declines (T1) as problems are found and solved during machine toning. Then for a significant period (T2), the machine remains relatively stable in terms of its failure rate performance, until the machine wearage becomes more serious and failure rate increases (T3).

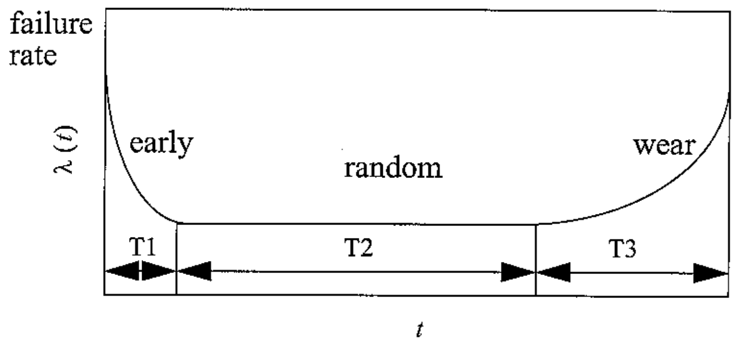


Figure 4. A Bathtub Curve for Machine Failure Rate vs. Time

What this tells us is that machine breakdowns can be parameterized by its failure rate. Each machine has its own life tub curve. The failure parameters are supplied by the machine manufacturer or it can be summarized based on data collected from test running.

For the period of T2 when machine performance is relatively steady, we can model the failure events using a certain random distribution. Two parameters are important in describing the machine failure characteristics:

- Mean Time Between Failures (MTBF);

- Failure Duration (FD). i.e., time to repair the failure machine.

These two parameters are modeled respectively as random distributions. They give us a general idea of the machine breakdown pattern. For instance, a drilling machine with a MTBF of $U(10, 2)$ and a FD of $U(1, 0.5)$ means that the machine normally breaks down once every 8 to 12 days. For each breakdown, it takes from half a day to a day and a half to put it back to operation. This representation is very helpful in an environment where machine failure happens regularly and frequently. so Preventive maintenance can be arranged accordingly. Other distributions commonly used to describe failures include exponential, normal and beta distributions [Hong 92].

The failure performance in T1 and T3 can also be modeled using MTBF and FD, except that both of them are functions of time. But they could still be approximated as constant during a short time interval.

The impact of machine failure on scheduling is primarily the chaos that it brings to the production floor. Shortage of capacity is a common result. This may be compensated by providing alternative resource capacity for the failure prone machine. Either a similar machine in the same machine group or another machine with processing flexibility (such as an automated machining center) can be supplied.

- ***Defective Materials***

Defective consumable materials can interrupt the production process. The production will not resume until enough materials that are up to the quality standard are available. The time to recover from the interruption depends on the safety stock level or ordering lead time.

One practical method is to keep a reasonable level of safety stocks. This applies to all materials, either that ordered from outside suppliers or that produced internally. Ordering lead time has to be taken into account in the decision making. Since both raw material and work in process inventory carry a large amount of investment and thus is unfavorable, it is not a good idea to keep as many inventory as possible. A typical practice is to keep the level of inventory that can last X days of production in the plant. X depends on the ordering lead time and all other factors that affect the inventory level.

Flexibility is helpful in case of material defects. Production can resume quickly if an alternative material is available or if it can be acquired in a short time. Otherwise if the current activity can be suspended and a replacement activity can be switched into, the machine/staff allocated for the suspended activity will not be wasted while waiting.

- ***Staff Smoothing***

Staff scheduling is very important for any labor intensive production, which is very common in the manufacturing industry. It is a relatively independent research issue from manufacturing scheduling. We are not going to address it in this thesis. The main difficulties lie in the fluctuation of demand versus the employment policies imposed by labor unions. Common approaches include employing a suitable mixture of full time and part time employees to cut the employment cost; carefully scheduling shifts to meet the varying demands during a period; cross training staffs so that staff absenteeism can be mitigated, and etc.

1.3.4 Human intervention - Verbal Expression

The computer scheduling system plays the role of decision support to the human scheduler under most circumstances. Wherever human intervention occurs, there are different individual elicitation of the same data or information. The user interacts with the scheduling system both in terms of input and output. Therefore the manner in which people represent and express uncertainty must be taken into account. In general, uncertainty management has to facilitate the reasoning about uncertainty and the communication to users of the results of that reasoning.

The manner in which people most frequently communicate uncertainty is through verbal uncertainty expressions. Those expressions commonly encountered in human being's language are: *definite, likely, unlikely, probable, impossible, doubted, unsure, unsuspected, infeasible*, and so on. The importance of incorporating uncertainty expressions into decision making is highlighted in expert systems, where human belief and knowledge are constantly consulted. Verbal uncertainty expressions are treated either as probability values [Cohen 89] , probability ranges, fuzzy probability intervals [Bonissone 87] , or simply as nominal annotations.

Several important notes have to be made before putting verbal uncertainty expressions into an application. First, there has to be a nice user interface for communicating with system users. Secondly, the users have to hold a consistent viewpoint in the meanings they assign to verbal uncertainty expressions. Third, if parameterized data can be used and does reflect the nature of the uncertainty in concern, verbal uncertainty expressions should be avoided for inconsistency and complexity.

1.3.5 Summary

The below table summarizes the types of uncertainty discussed above. We identify if the uncertainty type is modeled at the job shop scheduling level. We also prioritize the need to deal with them. The objective is to set a guideline for our first effort to deal with uncertainty in scheduling. It should be understood, though, that no single method may address them all. The remainder of the thesis will focus on dealing with machine failure uncertainty, which is of the highest priority in our list.

Table 1. Types of Uncertainty

Uncertainty Type	Classification	Schedule Level?	Approaches	priority
Demand uncertainty	statistical	N	Forecasting, inventory control, flexibility	
Supply uncertainty	non-statistical	N	Flexibility, safety stock	
Defective materials	none statistical	N	Inventory control	
Activity duration variation	objective	Y	Randomize, increase capacity	2
Alternative routing	objective	Y	Disjunctive process plans; resource pools	3
Resource Failure	objective	Y	Randomize, increase capacity	1
Staff uncertainty		Y	Shift schedule	
Human intervention	subjective	Y	Knowledge representation, expert system	4

1.4 Objective of Uncertainty Research in Scheduling

While scheduling tries to allocate time intervals for each activity on a particular resource, the uncertainties about the resource and the activity itself collaboratively deviate the reality from the schedule and gradually or abruptly invalidate the schedule generated.

Research in this area is trying to mitigate the impact of the uncertainty factors in scheduling, so that the cost of implementing the schedule could be decreased. The research has to answer at least the following questions:

- Identify the types of uncertainty associated with scheduling. What are the appropriate measurements to represent each of them?
- Which strategy should be taken, predictive or reactive? What uncertainty factors can be prepared for during schedule generation? What factors can only be reacted during execution?
- Is it feasible at all to generate a schedule which can be robustly executed in an uncertain environment? If yes, what are the associated cost? Can the problem be solved in a reasonable time?
- What are the measurements which distinguish a schedule's sensitivity to uncertainties.
- How to help the decision maker to foresee problems and provide protection?

To bring some insights into the answer of the above questions, the remainder of the thesis will extend and experiment a *Temporal Protection* approach [Chiang & Fox 90] which aims at providing robustness during predictive scheduling. Chapters three will introduce the concept of Temporal Protection. Chapters four and five will discuss the implementation in detail. This work is based on the belief that predictive scheduling plays an important role with respect to resource failure uncertainty. However, it is not intended to remove the need for reactive or iterative scheduling. We focus on researching the predictive scheduling approach instead of comparing it with reactive results.

1.5 Thesis Contributions

This thesis contributes in the following aspects:

1. We advocate the importance of preparing predictive schedule to achieve robustness in dealing with machine failure uncertainty. Temporal protection [Chiang & Fox 90] was used to provide a mechanism for describing machine failure uncertainty and protecting schedule accordingly, as well as reacting to reality during schedule execution.

2. The implementation of Temporal Protection in ODO:TNG allows the modelling and reasoning of machine failure uncertainty in the domain of constraint based scheduling.

3. We implement a simulation mechanism in ODO:TNG, providing a general tool for evaluating schedule robustness under stochastic failures.

4. For both single machine and job shop scheduling problems, we empirically demonstrate that temporally protected schedules deliver more robustness and incur less schedule execution cost than if the schedule is not protected. This work serves as an attempt to deal with uncertainty factors in job shop scheduling.

1.6 Summary of Remaining Chapters

Chapter 2 reviews related work in the area of scheduling under uncertainty. Both Operations Research and Constraint Based Scheduling approaches are reviewed. Chapter 3 introduces the concept of robust scheduling and Temporal Protection. Chapter 4 applies the method to single machine scheduling problems and evaluates some possible models of protection. Chapter 5 describes how constraint graph can be extended to incorporate Temporal Protection and other implementation details in ODO:TNG. Chapter 6 explains the experiments performed within ODO:TNG for evaluating the competency of the proposed model under job shop scheduling problems.·In Chapter 7 we conclude the work and outline some potential areas for future research. Appendix A1 to A4 help to better understand the general nature of uncertainty. Appendix A5 and A7 explain the ODO:TGN policy design and implementation environment. Appendices A8 contains 6 problem description files (in PODL) we used in experimenting ODO:TNG.

Chapter 2 Related Work

This chapter reviews the related research that is devoted to uncertainty management in the domain of scheduling.

2.1 Mathematical Programming Approach

Both deterministic and stochastic scheduling problems can be mathematically modelled. But it tends to be very difficult to solve, except for the case where the problem size is very small (number of machines and activities).

[Mittenthal 93] investigated the single machine scheduling problem. The objective was to minimize a nonregular penalty function when the machine may suffer random breakdowns. A related article by [Allahverdi 94] discussed the scheduling of n jobs on m parallel machines when the machines are subject to random breakdowns and job processing times are random variables, objective function being minimizing expected mean flow time. Both articles transformed the problems into deterministic models under some strict conditions. [Hong et al. 92] developed an analytical model for a production flow line with unreliable machines and random processing times. The behavior of the n-machine line is approximated by the behaviors of the (n-1) two-machine lines based on the decomposition. Simulation was done on a 3 machine line. The analytical model was reported to perform better than another method developed by [Choong 87]. An interesting situation called deterioration of processing times was described in [Mosheiov 94], where job processing time linearly deteriorates with its starting time. Single machine scheduling model was extended and algorithms are developed for preference over common performance measures: makespan, flow-time, total tardiness and number of tardy jobs. [Wein 91] investigated the impact of processing time knowledge on dynamic job shop scheduling. The job shops are modeled as multiclass networks of queues and a rule derived from a Brownian analysis of the network was applied. One deterministic and six stochastic scenarios were simulated and Brownian policy was found to treat unpredictable variability better. A recent research by [Leon 94] proposed robustness measures and embedded them in a generic algorithm to generate robust schedules to be implemented in job shops that are subject to machine disruptions. Makespan and job delay were performance measures for comparison. Their experiments advocated a 1.5% improvement on makespan and a 9.5% improvement on the expected job delay.

Probability is often used to quantify uncertainties. Analytical algorithms have been designed to achieve different objective function goals. But the computational complexity is high. Only very small-size problems (one or two machines) can be solved within a reasonable time. The size and complexity of the real-life scheduling problems rarely permit optimal schedule generation. This gives rise to the use of knowledge-guided search or heuristic search to deal with the complex domain specific interactions.

2.2 Human Interactions - Expert Systems

Expert system techniques have been successfully applied to the scheduling domain where human experiences is invaluable in the decision making process [Fox 84] . An experimental research by [Nakamura 88] even suggested that human decision making is superior to general dispatching rules in an interactive simulation environment that they created. [Masood 90] represented beliefs on the states of the world. Domain specific actions confronting various uncertainty situations are implemented. Beliefs are revised when new knowledge is obtained from the shop floor. Reactions to the events are deducted and executed. The expert system approach models the scheduling process by state space transitions [Shaw 87] . Both the 'uncertain data' and the 'uncertain knowledge' can be represented and reasoned. The realistic difficulty for ES approach is that human knowledge is sometimes hard to obtain, and updating the knowledge base can be quite a job. Also the knowledge is so domain specific that a general scheduler is hard to develop.

2.3 Hierarchical Scheduling

Scheduling can be tackled from two points of view: predictive and reactive. Although reactive scheduling methods can be easily executed in a dynamic and unpredictable world, they may formulate inefficient schedules. Predictive scheduling, on the other hand, generate complete schedules based upon estimated system states. The problem arises when domain uncertainties upset the estimates and interrupt the schedule. Many researches believe that a combination of prediction and reaction must be employed to effectively construct and implement a schedule [Bensana 93] [Foote 92] [Tam 94] [Ringer 90] . This combination is usually implemented in a multiple level environment, where there are different levels of abstraction based on the temporal distance from the execution time. What unexpected events can be reacted and how to react are domain specific. But the idea is to reduce the computational cost by abstracting the problem at different levels. The schedule will be constructed with less detail (or not constructed at all) be it further from the execution time of the schedule. Multi-level scheduling similar to that in Figure 5 is often used in this case [Foote 92] [Ringer 91]. The higher levels make predictive schedules while the lower levels react to domain uncertainties by local dispatch rules. The top two levels perform forecasting and planning. The bottom level corresponds to what we refer to as job shop scheduling. Note also that the impact of uncertainty on decision making has been investigated less extensively at the bottom level than at the other levels.

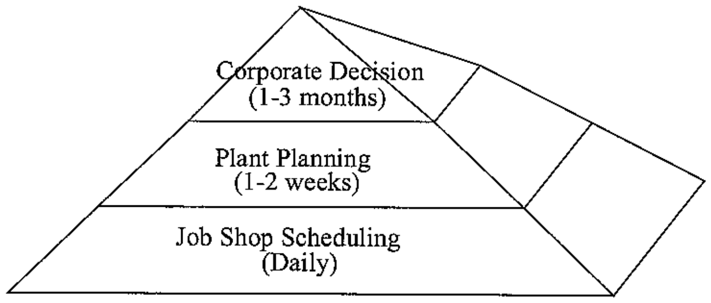


Figure 5. Hierarchical Decision Making Structure.

2.4 Constraint Based Scheduling (CBS)

Constraint based scheduling has been accepted as a viable AI-approach to scheduling problems [Fox 87] [Sadeh 91] [Zweben 94]. In this approach, the scheduling problem is represented by a constraint graph, where the nodes are the variables (activity execution times) and the arcs are the constraints (temporal, resource, or optimization constraints) between the variables. Solving the scheduling problem amounts to assigning values to all variables such that all constraints are satisfied. Different searching mechanisms, variable and value ordering heuristics are available for solving a scheduling problem represented as a constraint graph [Davis 94] [Sadeh 91].

Regardless of many successful research and application of CBS to job shop scheduling [Sadeh 94] [Saks 93], the issue of scheduling under uncertainty has not been sufficiently addressed, although the problem is constantly mentioned to be hard to solve. Classical constraint satisfaction relies on two assumptions: the problem is completely specified, and the constraints cannot be changed. Practical application challenges the assumptions by incomplete or imprecise problem description and dynamic changes to the activity network after the schedule is generated. Several efforts have been devoted to overcome the barriers.

An interesting application effort was accomplished by [Drummond 94]. A Just-In-Case scheduler called CERES, was produced to schedule and execute NASA's remotely located automatic telescopes under activity duration variance. A technique called contingent scheduling was used to explicitly consider the way in which actions might fail and how such failures can impact the desired schedule. It requires that stochastic actions are known in advance. By reasoning about possible execution errors in advance, the scheduler can create alternative schedules for the execution system.The 'dead time' of the observation period is greatly reduced. There is considerable amount of overhead incurred because of the inclusion of stochastic outcomes at each step. What is more, possible actions for the stochastic situations are not always known in advance.

There is the advocate that scheduling systems should only plan a few steps into the future and being fully reactive (or real-time). [Tadepalli 90] presented a real-time scheduling system called RTS. It takes partial scheduling actions after every constant time of real world, so that continual readjustment to the changing environment is possible (reaction to machine failure was not explicitly addressed in the paper). A fixed depth look-ahead search is performed from the initial state. A heuristic evaluation function is applied to the partial schedules at the leaves of the search tree to estimate the cost of the schedule. The system is thus taking the most promising scheduling action based on the evaluation function used. Experiments were reported to test the appropriate look-ahead depth for the search. A shallow depth was recommended as the result. However, truly real-time scheduler is not always possible for most scheduling environment except for the very low level of operations.

[Fargier 94] extended the classical constraint satisfaction problem to represent the qualitative or probabilistic uncertainties in agricultural planning. Based on the fuzzy nature of the domain constraints (the current cultivation action may have different consequences on crop behavior depend on unknown future events), satisfaction scale is attached to certain constraints and stored in the local knowledge base. Solving mechanism is a combination of classical backtracking and full lookahead to reach the maximum expectation of satisfaction. The paper also talked about the difference between flexibility and uncertainty. The method is novel but is hard to be applied to job shop scheduling area. No corresponding domain knowledge exists.

Chiang and Fox [Chiang & Fox 90] first introduced the concept of *Temporal Protection* to "prevent a predetermined schedule from temporal deviation caused by machine failures". A type-2 bound which is comprised of an inner bound (estimated mean processing time) and an outer bound (estimated maximum processing time) represents an activity's processing interval with protection allowance. Scheduling overlaps slack segments of consequent operations. Their experiments on single machine problems vary from 5 orders to 100 orders (one operation per order). Type-2 bound was compared to three other methods with fixed processing time. Their results showed that in general, type-2 method yields better performance in terms of total cost of work in process and tardiness. The paper suggested future work by allowing multiple operations within an order and using less reservation than the mean processing time. We found this approach promising in dealing with "known" machine uncertainty, although more experiments are needed for choosing the appropriate reservation bounds. The method also needs to be validated under a more complex scheduling environment than single machine. We choose to extend this temporal protection method in this thesis and perform a more thorough set of experiments both for single machine scheduling and job shop scheduling.

We have presented in this chapter the relevant work in the area of uncertainty management for scheduling. We have seen that the issue was researched from several perspectives. However, the relative strengths and weaknesses of these approaches remain to be assessed. It is clear that uncertainty management has not been researched extensively enough in the area of constraint based problem solving. No acknowledged method or benchmark data is available. More experiments need to be done to investigate the stochastic behaviors of uncertain events and their impacts on schedule generation and execution.

This thesis focuses on examining the uncertainties at the job shop scheduling level. Primarily, machine failure uncertainty is chosen to be addressed in this research. The remainder of the thesis will be devoted to proposing, implementing and experimenting the Temporal Protection technique that we believe can improve the robustness of a predictive schedule under a dynamic environment.

Chapter 3 Robust Scheduling

This chapter introduces the concept of robust scheduling and a Temporal Protection approach to reach the goal of scheduling robustly.

3.1 Predictive or Reactive?

In the domain of job shop manufacturing, machines inevitably malfunction, materials fail, or resources are not available when required. Of the many effects of such events, schedule disruptions are perhaps the most visible ramifications. A disrupted schedule incurs higher costs due to missed customer delivery dates, higher work-in-process inventory, and idling of people or machines [Chiang & Fox 90].

It is obvious that as uncertainty increases in a scheduling domain, it is less likely that a predictive schedule will be implemented successfully and it is more likely that there will be a greater reliance on a reactive scheduler to generate an appropriate response [Ow 88] . Consequently, spending more time on developing precise schedules that further optimize a set of measures, such as reducing work-in-process or tardiness, may be unnecessary if the temporal granularity of the impact of uncertainty exceeds the granularity of the optimization of the schedule. But given the necessity of developing predictive schedules, in order to plan resource purchases, allocations and releases, the question arises as to how precise should temporal decisions be, and the nature of the flexibility that a reactive scheduler should be afforded in responding to stochastic events.

Figure 6 shows the contrasts between preparing for uncertainty predictively, and reacting to interruptions reactively. The predictive approach incorporates the knowledge of shop floor uncertainty (such as machine failure) into the schedule generating stage. The resulting schedule is thus more robust in its handling of interruptions. Since only "known" uncertainty factors can be prepared for, protection may be limited.

Reactive scheduling takes action only when uncertain events (such as a machine breakdown) actually occur and interrupt the execution of a schedule. Rescheduling will be attempted if the original schedule can not be implemented as planned.

In evaluating the two approaches, the question arises as to how much freedom should be given to each method? It is crucial to identify the nature of uncertainty in a schedule environment. When uncertainty can be mathematically modelled, a predictive approach can reduce the cost of rescheduling by taking uncertainty into account in schedule generation. For "unknown" uncertainty, a reactive approach may be more appropriate. This thesis focuses on the "known" (can be modelled mathematically) uncertainty. The goal is to create a robust predictive schedule which is less costly than pure reactive schedule when being executed.

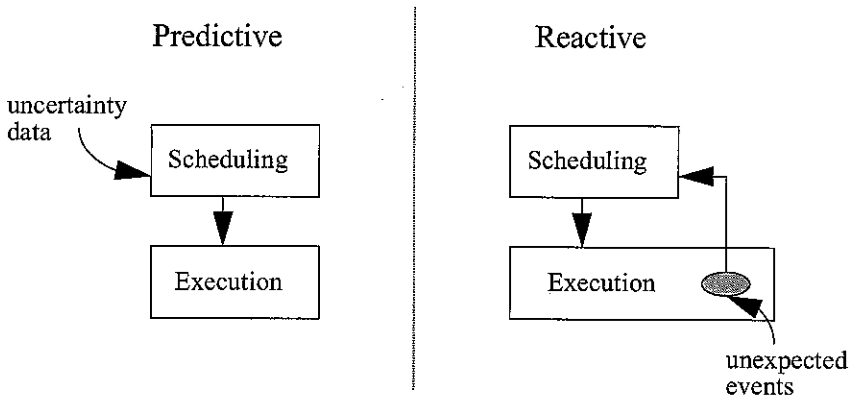


Figure 6. Attitudes in dealing with uncertainty: predictive or reactive?

3.2 Robustness of Schedules

3.2.1 What is Robustness?

An important issue in the relationship between predictive and reactive scheduling is that of schedule **robustness**. A robust predictive schedule is one which is likely to remain valid under a wide variety of disturbances [Leon 94] . Robustness of a predictive schedule will reduce the number of subsequent reactive scheduling decisions required and is usually desirable.

Robust scheduling tries to protect the schedule from being interrupted by stochastic events. The goal is to make robust schedules that incur less implementation cost (WIP cost, tardiness penalty, rescheduling time delay, etc.) than simple reactive scheduling. When a stochastic event occurs, the original schedule can be adhered to as much as possible. The point is that robust schedules are prepared for the possible occurrences, whether they happen or not.

In the Constraint Based Scheduling approach, the scheduling problem is represented as an activity network with constraints linking variables. The task of scheduling is to assign values to the variables so that the constraints are satisfied (network consistency achieved). Uncertainty events reflect dynamic changes to the network after scheduling, such as removing or inserting a constraint. A robust network has the characteristic of absorbing the changes locally so that the effect is not propagated through the network.The robustness that should be provided to an activity network depends on the level of uncertainty that is present in its environment.

3.2.2 Measuring Robustness

A schedule's robustness can be measured by two criteria. One is the deviation of the schedule execution from the original schedule. The other is the real cost incurred by the implementation of the schedule.

- Deviation from the original schedule:

- maximum deviation for activity planned start time from real start time.
- difference between planned makespan and actual makespan.

- The cost of schedule execution may include:
  - work in process (WIP) cost for all activities. Assuming C_w as the unit cost for wip,

$$WIP = \sum_{all-activities} C_w \cdot (endTime - releaseTime) \ .$$

  - tardiness penalty. Assuming C_t as the unit cost for tardiness,

$$Tardiness = \sum_{all-activities} C_t \cdot (RealEndTime - DueDate)^{+\ 1} .$$

  - resource idleness cost due to late release. Assuming C_i as the unit cost for resource idleness,

$$Idleness = \sum_{all-activities} C_i \cdot (ReleaseTime - EarliestPossibleStartTime)^{+} \ .$$

Our later experiments take the three costs of schedule execution as the basis for comparison. The deviation criteria is worth looking at for understanding the dynamics of schedule execution.

3.3 Temporal Protection

We investigate the predictive approach to creating robust schedules. We believe that the predictive approach is preferred when the uncertainty characteristics can be mathematically modelled and the model incorporated into the process of schedule generation. Our method of achieving schedule robustness is to provide *temporal protection* (will be explained in the next section) when building a schedule.

3.3.1 Activity Reservation

When we say an activity is scheduled, it means resources are reserved for a time period for this activity. In job shop scheduling, an activity often requires more than one resource to perform the operation. Typical ones include machine(s), an operator, tools (fixtures, cutting tool, etc.) and raw materials (parts, cooling liquid, etc.). While machine availability depends on the status of shop floor execution, other resources can be scheduled to release at a predefined time. We consider the case when only machines are failure prone. Failure prone machines are referred to as *critical* resources in this thesis, versus *non-critical* resources that are reliable during scheduling window.

A traditional reservation for an activity is like that in Figure 7. In this case, all resources for this activity are allocated for the activity's duration. All resources are scheduled to be released at the activity start time. If the critical machine is available at start time, the activity can start as scheduled. But if the machine fails, the activity's actual duration is longer than scheduled. This violation is propagated to subsequent activities, whose released (non-critical) resources are left waiting for the machine to become free. Activities following this one will incur more wip cost on the non-critical resources, though machine idleness is minimized.

1. $X^{+} = 0, (X < 0), X^{+} = X, (X \geq 0)$

Figure 8 shows the reservation bounds of two sequential activities under *Temporal Protection* originally proposed by [Chiang & Fox 90]. A reservation is composed of an *inner* interval P_{inner} and an *outer* interval P_{outer}. P_{inner} defines the start time and estimated duration of the activity, i.e., the time during which we expect the activity to be performed. P_{outer} defines the earliest time we would expect the activity to start. Both P_{inner} and P_{outer} are functions of the original duration and the knowledge about the resource failures. The modifications of the activity's duration and resources release time are referred to as duration protection and release time protection respectively. The overlapped slack intervals between the activities provide the release flexibility during schedule execution. It struck us that the usage of the upper slack would cause extra tardiness, since the subsequent activity is pushed later by the amount of upper slack. We therefore revised the representation into that shown in Figure 9, where we combined the two slack intervals into one lower interval. i.e., while two bounds remain the same, the release time protection is strengthened. Note that resources are released differently under this representation. The critical resource is allocated for the period of P_{inner}. Non-critical resources for the same activity are released earlier at *release-time*. When executing the schedule, if the previous activity actually takes less time than the protected duration, the critical resource is released earlier than 'start' for the current activity. The activity can thus start any time once the critical machine is available (as early as release-time).

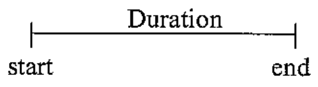


Figure 7. Normal Reservation Bounds.

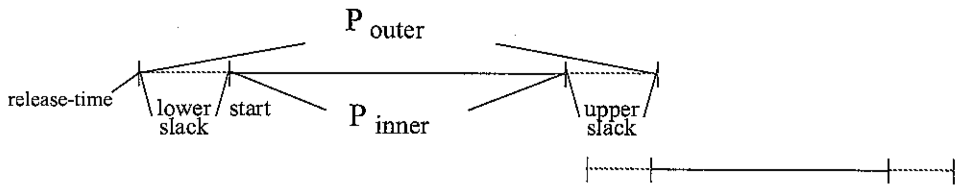


Figure 8. Reservation Bounds Used in [Chiang & Fox 90].

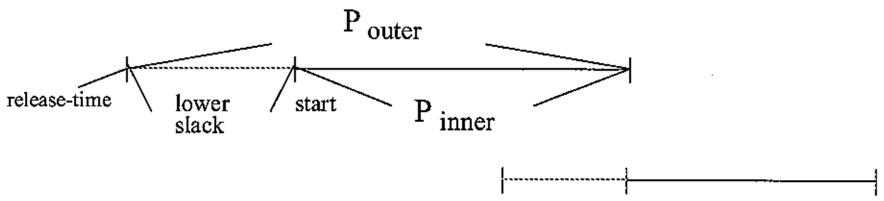


Figure 9. Reservation Bounds as We Propose.

3.3.2 Scheduling with Temporal Protection

In scheduling with our revised temporal protection model, the middle segment (P_{inner}) is necessary to reserve the resources for the activity while the lower segment $(P_{lower-slack})$ may be overlapped with the P_{inner} intervals of other activities. The start point of $P_{lower-slack}$ defines the release time for non-critical resources. In Figure 10, activity A is designed to be overlapped with activity B. If A completes earlier, then B may start earlier to the extent of its lower slack. When uncertainty increases, the precision of this predictive schedule is decreased and the lower slack is larger. This overlapping gives the dispatcher the flexibility to start the job at any time after the release time. At the same time, P_{inner} is further extended when uncertainty increases, providing more protection during the execution of the activity.

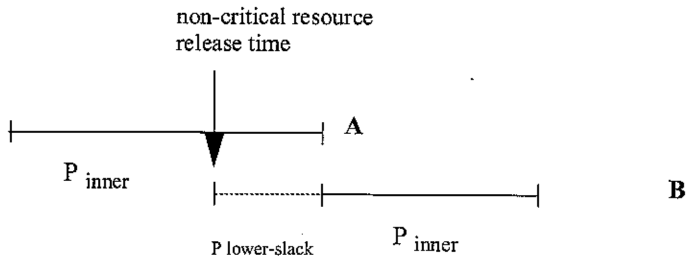
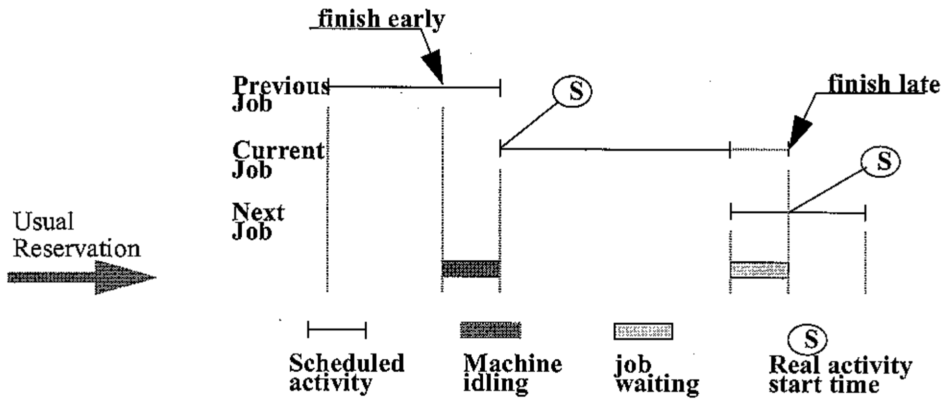


Figure 10. Illustration for two overlapped activities

The difference between using our temporal protection as activity reservations in a schedule and using the usual definition of activity reservations can be seen in Figure 11. The old reservation is simply a time interval with sharp start and finish times. Our definition of reservation adds a lower slack interval offering the flexibility to start an activity any time after its release time.



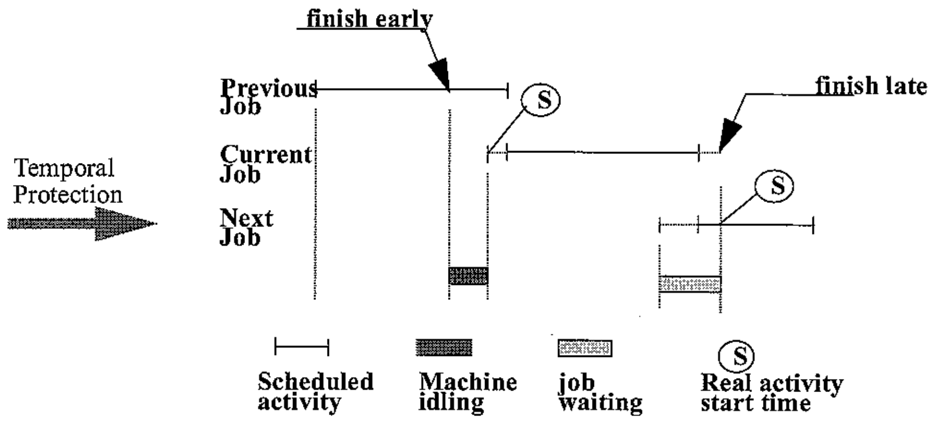


Figure 11. Scheduling with usual reservation vs. with temporal protection

Chapter 4 Experimenting Temporal Protection on Single Machine Scheduling Problems

This chapter extends the work of [Chiang & Fox 90]. Variations of the original temporal protection calculation model are tested to validate the model of choice. Experiment design is described and the results are analyzed.

4.1 Calculating Temporal Protection

Confronted by resource failure uncertainty, an obvious way is to extend the duration of each activity requiring that resource, although the amount of extension needs to be empirically proved. Temporal protection specifies at what time an activity's materials are to be released to the factory, and how much temporal slack is to be supplied to the activity's duration. The following denotations originated from [Chiang & Fox 90]. We will show how temporal protection is derived and used in the specification of an activity's reservation for resources. These will be determined by forecasting an activity's earliest start and latest end times under possible machine breakdowns.

Let the original processing time of an activity be P, which is deterministic in the model, the time between machine failure be a random variable F, and the duration of interruption be a random variable D. We consider two methods to express the processing time to include the machine interruptions. Let these times be called extended processing times P_{ext1} and P_{ext2}.

P_{ext1} (see EQ1) accounts for the extended processing time by adding the total expected down time to the uninterrupted processing duration. P/F gives the random number of breakdowns during the processing of a activity. P/F multiplied by D is the random variable that represents the duration extension caused by the machine breakdowns.

$$P_{ext1} = P + \frac{P}{F} \times D \qquad \textbf{(EQ 1)}$$

If the mean of D and F are known as $\bar{F}$ and $\bar{D}$, then the mean of P_{ext1} is given as $P + \frac{P}{\bar{F}} \times \bar{D}$. Let this be P_{mean-1} (see EQ2). $\frac{P}{\bar{F}}$ gives the mean number of interrupts that may occur during the processing time and $\frac{P}{\bar{F}} \times \bar{D}$ gives the mean total length of the machine down time.

$$P_{mean-1} = P + \frac{P}{\bar{F}} \times \bar{D} \qquad \textbf{(EQ 2)}$$

P_{ext1} applies when we interpret Mean Time Between Failures (MTBF) as the average of total up time divided by the number of failures. If MTBF is defined as the whole time interval divided by the number of failures, then additional down time in the extended durations should be accounted for. P_{ext2} is the total extended processing time under this situation. P_{ext2} instead of P is used to calculate the expected number of interruptions (see EQ3). P_{ext2} is formulated in EQ4 and rewritten in EQ5. Similar to P_{mean-1}, we denote the mean of P_{ext2} as P_{mean-2} (see EQ6).

$$Expected-number-of-interruptions = \frac{P_{ext2}}{F} \qquad \textbf{(EQ 3)}$$

$$P_{ext2} = P + \frac{P_{ext2}}{F} \times D \qquad \textbf{(EQ 4)}$$

$$P_{ext2} = \frac{P}{(1 - D/F)} \qquad \textbf{(EQ 5)}$$

$$P_{mean-2} = \frac{P}{(1 - \overline{D}/\overline{F})} \qquad \textbf{(EQ 6)}$$

Instead of being random variables of known distribution, the failure duration and the time between failures may be only known to be bounded approximately. Let the bounds be (D_{lb}, D_{ub}) for D and (F_{lb}, F_{ub}) for F with the means $\overline{D}$ and $\overline{F}$, we can therefore determine the bounds of extended processing time using the following interval arithmetic [Kaufmann 84] . Let A be an interval bounded number with upper and lower bounds defining an interval noted as $[a_1,a_2]$ where $a_1 \le a_2$. Similarly, let B be a number associated with an interval $[b_1,b_2]$ where $b_1 \le b_2$.

- **Addition:** If $x \in [a_1,a_2]$ and $y \in [b_1,b_2]$, then $x + y \in [a_1 + b_1, a_2 + b_2]$ is also a bounded number. Symbolically write it as $A(+)B = [a_1,a_2] + [b_1,b_2] = [a_1 + b_1, a_2 + b_2]$.
- **Subtraction:** Similarly, $A(-)B = [a_1 - b_2, a_2 - b_1]$.
- **Multiplication:** Similarly, $A(*)B = [a_1 \times b_1, a_2 \times b_2]$
- **Division:** Similarly, $A(/)B = [a_1/b_2, a_2/b_1]$.

Given the above interval arithmetic, bounds of P_{ext1} and P_{ext2} can be calculated. Since uninterrupted processing time P is a fixed value, the bounds of extended durations depend exclusively on the lower and upper bounds of D and F. The lower and upper bounds for $D(/)F$ are given in EQ7 and EQ8 respectively. The lower and upper bounds for P_{ext1} and P_{ext2} (denoted by adding *-lb* or *-ub* to the original subscripts) are given in EQ9 through EQ12:

$$(D(/)F)_{lb} = D_{lb}/F_{ub} \qquad \textbf{(EQ 7)}$$

$$(D(/)F)_{ub} = D_{ub}/F_{lb} \qquad \textbf{(EQ 8)}$$

$$P_{ext1-lb} = P + P \times D_{lb}/F_{ub} \qquad \textbf{(EQ 9)}$$

$$P_{ext1-ub} = P + P \times D_{ub}/F_{lb} \qquad \textbf{(EQ 10)}$$

$$P_{ext2-lb} = P/(1 - D_{lb}/F_{ub}) \qquad \textbf{(EQ 11)}$$

$$P_{ext2-ub} = P/(1 - D_{ub}/F_{lb}) \qquad \textbf{(EQ 12)}$$

The failure statistics for D and F may originate from subjectively known machine characteristics described by the manufacturer. Shop floor operational statistics may be used to update the knowledge when the machine gets older. Our next task is to specify the temporal properties of an activity's reservation. As stated earlier, there are two properties of importance, the scheduled duration for the activity and the resource release time (i.e., expected earliest start time). The next section defines several models for determining the two properties.

4.2 Models of Temporal Protection

We explored four methods, TP1-4, for calculating the inner and outer bounds as activity reservation was defined in chapter 3. The objective is to find a suitable amount of protection based on our knowledge of machine failure. This selection is important because too little protection will not provide enough robustness to the schedule and too much protection will waste resources.

TP1: is the original protection model from [Chiang & Fox], but the lower slack is defined differently and there is no upper slack. It uses average duration extension P_{mean-1} as the protected activity duration. Longest estimate duration $P_{ext1-ub}$ is used as the outer interval. Slack interval is the difference between the two. Based on our interpretation of mean time between failures (down time is not included in it), we expect this model to reflect the "state of the world" and thus perform better than other variations listed below. Calculation steps are:

- Define mean of P_{ext1} (see EQ2) as P_{inner}: $P_{inner} = P_{mean-1}$
- Define upperbound of P_{ext1} (see EQ10) as P_{outer}: $P_{outer} = P_{ext1-ub} = P + P \times D_{ub}/F_{lb}$
- $P_{lower-slack}$ is half the difference between P_{outer} and P_{inner} : $P_{lower-slack} = (P_{outer} - P_{inner})$

TP2:is a variation of TP1. It uses the same P_{mean-1} as the protected activity duration. Try outer interval as P_{mean-2} to test if the assumption of extra delay works with the maximum estimate processing time. The intention is to see the difference between TP1 and TP2 due to this outer bound change, which impacts release time only. Calculated as:

- Define mean of P_{ext1} as P_{inner}: $P_{inner} = P_{mean-1}$
- Use the difference between P_{mean-2} (see EQ6) and P_{mean-1} as $P_{lower-slack}$, set $P_{lower-slack} = P_{mean-2} - P_{mean-1}$.

TP3: Varied from TP2, but change the duration protection definition this time. Assume P_{ext2} has a probability density function of $f(P_{ext2})$ (Figure 12 shows a triangle distribution of P_{ext2}) on its bounded interval $[P_{ext2-lb}, P_{ext2-ub}]$. Use the expected mean of P_{ext2} as P_{inner} to see the amount of deviation caused by using a different estimate than our failure scenario (down time not included in MTBF). P_{mean-2} remains as outer interval. Calculation follows:

- Calculate the expected value of P_{ext2} between $P_{ext2-lb}$ and P_{mean-2}. Denote this to be: $P_{exp-lower} = \int_a P_{ext2} \cdot f(P_{ext2}) dP_{ext2}$ where $a = P_{ext2-lb}$ and $b = P_{mean-2}$. Let $P_{inner} = P_{exp-lower}$.
- Set $P_{lower-slack} = P_{mean-2} - P_{inner}$

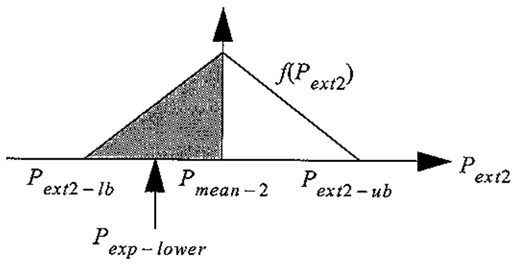


Figure 12. Example probabilistic density function for P_{ext2}.

TP4: We design TP4 to have the most restricted duration protection $P_{ext1-lb}$. It is chosen to test the impact of less protection than P_{mean-1} as in TP1. Outer bound is the same as in TP1, i.e., $P_{ext1-ub}$. Calculated as:

- As defined in EQ9 and EQ10: $P_{ext1-ub} = P + P \times D_{ub}/F_{lb}$, $P_{ext1-lb} = P + P \times D_{lb}/F_{ub}$
- Define $P_{inner} = P_{ext1-lb}$, $P_{outer} = P_{ext1-ub}$.
- Use half the difference between $P_{ext1-ub}$ and $P_{ext1-lb}$ as $P_{lower-slack}$: $P_{lower-slack} = (P_{ext1-ub} - P_{ext1-lb})$

A numerical example can be used to illustrate the calculations. In the example, bounds and means of mean time between failures (F) and failure duration (D) are given as in Table 2. All time units are in minutes.

Table 2. Machine Failure Data

(D_{lb}, D_{ub})	$\bar{D}$	(F_{lb}, F_{ub})	$\bar{F}$	P
(4,6)	5	(15, 25)	20	40

The protection allowance for the four temporal protection methods are calculated. The results can be visualized as in Figure 13.

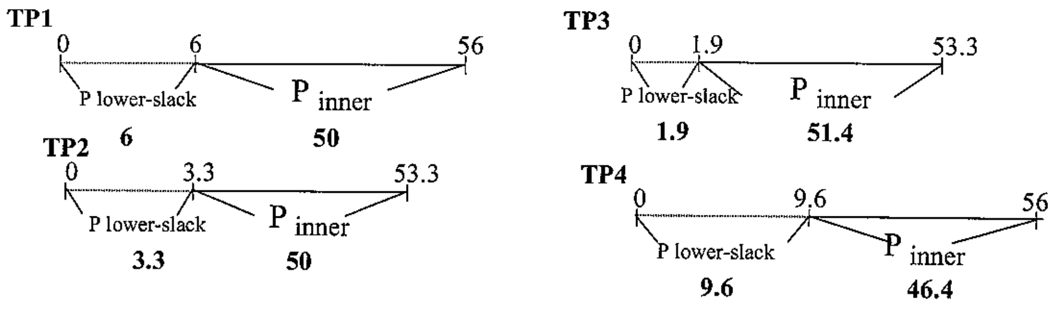


Figure 13. Numerical example for the temporal protection bounds

4.3 Experiment Design

4.3.1 Methods Grouping

In order to measure the effectiveness of temporal protection, we compare eight approaches to managing uncertainty. The eight methods can be grouped into three groups (as shown below). The first method alone comprises group one. It uses the original processing time in scheduling, providing no temporal protection at all. It is generated to see what if nothing is done. The second through fourth methods offer duration extension but no release time flexibility and will be our baseline for comparison to the next group. The fifth through eighth methods are TP1 to TP4 as described in section four. They provide protection for both activity duration and release times. Numerical experiments are designed to test their effectiveness in dealing with machine failure uncertainty in single machine scheduling.

Group1: no protection at all.

1. **no-protection:** using the prescribed processing time P as the fixed processing time. Just a check to see what happens if nothing is done.

Group2: duration extension protection.

2. **mean method-1:** using P_{mean-1} (EQ2) as the fixed processing time. Expect good performance since it represents average processing time estimate. An intuitive way of protection. Will be used as our baseline for comparison.
3. **upperbound method:** using $P_{ext1-ub}$ (EQ10) as the fixed processing time. Used to test a more relaxed duration protection than mean-1.
4. **mean method-2:** using P_{mean-2} (EQ6) as the fixed processing time.

Group3: duration protection plus release time protection.

5. **TP1** (see section 4.0 for definition). Use **mean method-1** as the baseline to compare. Same duration protection is used. Extra release protection is applied.
6. **TP2** (see section 4.0 for definition). Use **mean method-1** as the baseline to compare. Same duration protection is used. Extra release protection is applied.
7. **TP3** (see section 4.0 for definition). Use **mean method-2** as the baseline to compare. Similar duration protection is used. Extra release protection is applied.
8. **TP4** (see section 4.0 for definition). A test of very restricted duration protection. Expected to generate big WIP cost.

4.3.2 Problem Description

The environment contains one machine and 100 activities with requested due dates and release times. A predictive schedule is needed for resource acquisition and is fulfilled by a dispatcher which does not change the sequence of the input activities. There is neither preemption among activities. Machine fails from time to time. The distributions of mean time between failure and failure duration are known beforehand. If an activity is interrupted by a failure, it waits until the machine recovers and completes its processing. We will find out, by our experiments, if protection is needed when creating the schedule, and which protection model provides the most robustness for reacting to uncertainty.

Following are detail experiment data designs:

- **Activities**:

  - *Processing time*: Each activity has a processing time generated by a random variable with a distribution of U[100, 300], where U[] is a uniform distribution within a closed interval.

  - *Due date*: Four types of distributions were used for the due dates, representing different levels of tightness[1]. Each activity is assigned a due date generated by one of the four distributions.

Table 3. Single Machine Experiment - Due Date Tightness

Very Tight	Tight	Medium Loose	Loose
U(15000, 25000)	U(15000, 30000)	U(15000, 35000)	U(15000, 45000)

- **Machine Failures**

  - *Mean Time Between Failures* (*MTBF*): Four triangular distributions are used to describe MTBF. T(min, mean, max) depicts a triangular distribution over [min, max] with mean as the peak. The four distributions represent the cases when average activity duration P ($P = 200$ from $P \in U(100, 300)$) is bigger, equal, less than or far less than the mean of MTBF, i.e., we are investigating the impact where an activity is frequently interrupted, sometimes interrupted, or hardly interrupted.

Table 4. Single Machine Experiment - Mean Time Between Failures

MTBF1	MTBF2	MTBF3	MTBF4
T(80, 100, 120)	T(160, 200, 240)	T(320, 400, 480)	T(480, 600, 720)

  - *Duration of failures* (*D*): For each of the four distribution of *MTBF*, four distributions of *D* (D1 to D4) were generated, so that the ratio D/F = 0.01, 0.06, 0.16 and 0.31. For example, *MTBF2-D1* means $F = T(160, 200, 240)$ and $D = T(1.6, 2, 2.4)$. i.e., the duration of failure is another random variable which follows the scaled triangle distribution. Sixteen combinations of *MTBF* and *D* are investigated in the experiments.

1. Depending on frequency and duration of machine failures, the due date could be tight under one machine but loose under another.

- **Number of Observations:** Each of the three experiment parameters: due date tightness, machine failure frequency and failure duration, has four levels of values. Sixty-four experiments representing all combinations of activity data and machine failure data can be performed. For each setting, we randomly generated five scheduling problems each composed of 100 activities. Five observations were collected and analyzed accordingly.

4.3.3 Create Predictive Schedule

[Chiang & Fox 90] used Earliest Due Date (EDD) dispatch rule to create the predictive schedule. We added a second dispatch rule, ShortestProcessingTime(SPT), which minimizes the total flow time. See below for detail description of the EDD algorithm for n-activity single machine (no failure) problem, L_{max} (maximum lateness) is minimized by the sequence of EDD. SPT algorithm is the same except that in step 2, shortest d_i should be chosen and ties are broken based on the earliest b_i.

Jackson's Algorithm

At each activity completion, the activity with the minimum due date among unexecuted activities is selected to begin processing.

Let S be the set of unscheduled activities, b_i be the due date, and d_i be the duration of activity i. Schedule each activity as follows:

1. Set t to 0.

2. Among all activities $i \in S$, choose the activity j that has the earliest due date b_j; break ties on due date by selecting the activity with the largest duration d_i. Remove activity j from S.

3. Schedule the chosen activity to be performed at time t and update t to t+d_j,

4. If S is not empty, go to 2. Otherwise, the schedule is completed.

SPT Dispatching Rule

At each activity completion, the activity with the shortest processing time among unexecuted activities is selected to begin processing.

The output of schedule creation is a set of *planned release times* for each activity. In using the algorithms with temporally protected durations whose lower slack interval overlaps with previous activity, the resource release time has to be assigned as well. The algorithm is modified as below. Same modification applies to SPT scheduling and is not reiterated here.

Modified Jackson's Algorithm

Let S be the set of unscheduled activities, b_i be the due date, and d_i be the duration of activity i. P^j_{inner} is the protected duration and $P^j_{lower-slack}$ is its release interval. Schedule each activity as follows:

1. Set t to 0.

2. Among all activities $j \in S$, choose the activity j that has the earliest due date b_j; break ties on due date by selecting the activity with the largest duration P^j_{inner}. Remove activity j from S.

3. Schedule activity j to start at time t. Non-critical resource release time is set at $t - P^j_{lower-slack}$. Update t to $t + P^j_{inner}$.

4. If S is not empty, go to 2. Otherwise, the schedule is completed.

4.3.4 Simulate Schedule Execution

This simulation algorithm is based on that in [Chiang & Fox 90]. The schedule execution algorithm randomly generates machine failures during the execution of the schedule, thereby delaying the start of activities, or prolonging the execution of activities. An activity cannot start before its *planned release times*, even if the machine is available. The algorithm is defined as below.

Schedule Execution Algorithm

Let S be the set of scheduled activities, and a_i be the arrival time, b_i be the due date, d_i be the original uninterrupted duration, and rt_i be the planned release time of activity i. Schedule each activity as follows:

```
1. Set $t = 0$ and machine state to free.
2. Randomly schedule a machine failure and duration.
3. If the next machine failure time is less than the earliest planned released time of the remaining unexecuted activities, then set $t$ to the next machine failure time plus its duration, update machine idle time, go to 2.
4. If the machine state is free, and there is at least one activity planned to be released earlier than the next failure time,
    4.1. Choose activity $i$ with the earliest planned released time $rt_i$, remove it from $S$.
    4.2. If the current $t$ is less than $rt_i$, then set $t = rt_i$. Machine remains idle until $rt_i$.
    4.3. activity $i$ actually starts at $t$. Set the completion time $ct = t + d_i$.
    4.4. Set the machine state to busy.
5. If the machine state is busy, and the next machine failure occurs before the completion time $ct$, then add the machine failure's duration to the completion time, and go to 2.
6. Set $t = ct$, activity $i$ actually ends at $t$. Set the machine state to free, and go to 2.
```

4.3.5 Measuring the Costs

Tardiness and work-in-process are two statistics used in schedule evaluation. We define:

X^+: X=0 if X<0; X=X otherwise

- Tardiness = $\sum_{alljobs}$ (actual finish time - requested due date)$^+$

  where requested due date is the one generated randomly prior to scheduling (customer order due date). Another definition of tardiness is probably a preferred measurement to the dispatcher:

  Tardiness = $\sum_{alljobs}$ (actual finish time - scheduled finish time)$^+$

  For this experiment, we will use the first definition for tardiness from the customer's point of view.

- Work-in-process = $\sum_{alljobs}$ (actual finish time - planned release time)

  where planned release time is the result of predictive schedule. It is obvious that work in process will grow tremendously under frequent and long failures if we do not include failure protection in our schedule.

We understand that neither EDD nor SPT dispatch rule optimizes the above criteria. The intention is to compare the performance of our eight suggested methods, given the same dispatch rule and cost of concern. We will show that in the result analysis section.

Protecting against the effects of uncertainty introduces a third cost component: machine idling. Figure 14 shows the case when an activity finishes before any other activities can be started. The machine is left idle until the next activity can be started.

- Idleness = $\sum$ (planned release time - actual finish time of previous activity)$^+$

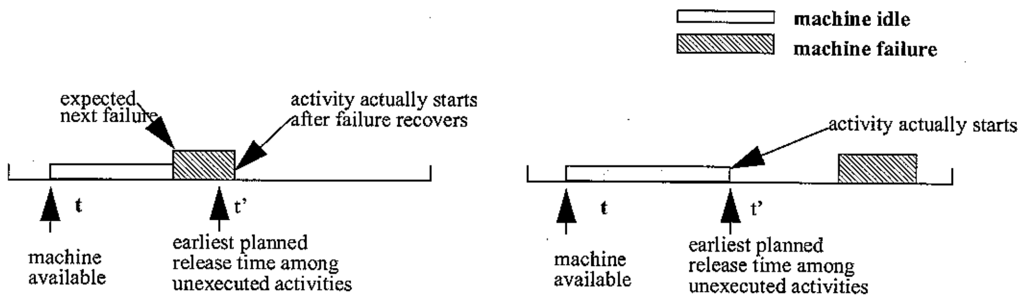

machine idle
machine failure
expected next failure
activity actually starts after failure recovers
t
t'
machine available
earliest planned release time among unexecuted activities
activity actually starts
t
t'
machine available
earliest planned release time among unexecuted activities

Figure 14. Machine idle due to failures or early activity finishing.

In the next section we compare the eight methods afore mentioned according to these three dimensions. In order to ascertain the context in which one form of temporal protection outperforms another, we compare them on a cost basis, where

$$TotalCost = C_T \times Tardiness + C_W \times WorkInProcess + C_I \times Idleness \qquad \text{(EQ 13)}$$

Where C_T, C_W and C_I are unit cost of tardiness, work-in-process and idleness, respectively. Seven sets of unit costs are used as in the below table, representing the varied weights of each type of cost.

Table 5. Different Cost Structures:

Cost Structure	1	2	3	4	5	6	7
C_T	1	1	1	1	2	2	2
C_W	1	1	2	2	1	1	2
C_I	1	2	1	2	1	2	1

4.3.6 Experiment steps

All experiments are composed of three steps:

1) For each experiment, a schedule was created according to one of the eight methods. One of two dispatching rules is employed: Jackson's algorithm, which minimizes the maximum lateness [Baker 74] and total tardiness cost, or SPT (Shortest Processing Time) dispatching rule.

2) The schedule was implemented using discrete event simulation in which resource failures occurred. For each experiment, machine failures were generated as random numbers following certain distributions.

3) For each run, data on tardiness, work-in-process and resource idleness were gathered.

4.4 Result analysis

The experiment described above is an example of Factorial design [Law 91] [Montgomery 91], which involves several factors where it is necessary to study the joint effect of the factors on a response. The factors employed in our experiments are simulation run numbers (5 levels), scheduling methods (8 levels), due date tightness (4 levels), mean time between failures (4 levels), ratio of mean duration to MTBF (4 levels) and unit cost types (7 levels). The number of experiment runs is the product of all the levels. There are altogether:
$5 \times 8 \times 4 \times 4 \times 4 \times 7 = 17920$ experiments. Data is analyzed by ANOVA[1] (ANalysis Of VAriances) application available on most UNIX environment. Statistical comparison on average total costs for EDD and SPT dispatching rules are shown in Table 6 and 7 respectively.

1. ANalysis Of VAriances: ANOVA does multi-factor analysis of variance on designs with within groups factors, between groups factors, or both. The input to anova consists of each datum on a separate line, preceded by a list of index labels, one for each factor, that specifies the level of each factor at which that datum was obtained. The output from anova includes summary statistics for each source factor. The summary statistics include: cell counts, means, standard deviations, and standard errors.A summary of design information and an F table testing main effects and interactions of factors are also generated. Sums of squares, degrees of freedom, mean squares, F ratio and significance level are reported for each F test. For significance testing, one asterisk indicates a result significant at the .05 level, with a confidence interval of 95%. Two *'s indicate .01 significance, and three *'s indicate .001 significance.

For both of the dispatching algorithms, use of temporal protection in scheduling significantly reduces the total cost from their no-protection cases. This manifests that leaving the schedule unprotected in front of failure uncertainty is not a good idea. We found that protected schedules (Groups 2 and 3) always perform much better than if the schedule was unprotected. Less total cost of work in process, job tardiness and machine idleness is incurred when the protected schedule is executed on failure prone machines.

Group 2 does significantly reduce costs as we expected. Extending duration is an effective way of dealing with machine failures. We also see that mean-1 in group 2 gives the best performance with an average estimate of processing time. The other two estimates (upperbound of mean-1 and mean-2) does not work as well as mean-1 in our uncertainty environment. We will use mean-1 as the baseline to compare other methods. The '% saving' columns in table 6 and 7 shows the percentage difference of each method compared to no-protection and mean-1 respectively. Numbers in () means an increase in cost.

From Table 6, we can see that mean-1 is a good performer (as we expected), while its extensions TP1 and TP2 each save the cost by 2% more. The extra saving is not obvious in the case of SPT (table 7). But TP1 still results in 1% extra saving than mean-1. All other methods increase the total cost from mean-1 under both EDD and SPT, manifesting their improper amount of duration protection. TP4 performs especially bad in both cases, showing the disastrous impact of not protecting enough. Look at TP1 and TP2 versus mean-1, and TP3 versus mean-2 in both tables, we can conclude that protecting both duration and release time is better than the intuitive duration extension as done by group 2 methods.

Table 6. Average of total costs - EDD

Group	**Method**	**N**[a]	**MEAN**[b]	**% saving (G1)**	**% saving (mean-1)**
G1	no-protection	2,240	229, 518		
	mean-1	2,240	54, 590	76.2%	
G2	upperbd	2,240	68, 507	70.1%	(25.5%)
	mean-2	2,240	64, 163	72.0%	(17.5%)
	TP1	2,240	53, 620	76.6%	2.0%
G3	TP2	2,240	53, 567	76.6%	2.0%
	TP3	2,240	58, 939	74.3%	(8.0%)
	TP4	2,240	112, 169	51.1%	(105%)
$\overline{(G2)}$		6,720	62, 420	72.8%	(14.3%)
$\overline{(G3)}$[c]		6,720	55, 375	75.9%	(1.4%)
$\overline{(G2,G3)}$		15,680	58, 898	74.3%	(8.0%)

a. Number of experiment runs.

b. Average total cost over all experiment runs using this method.

c. Average on TP1 to TP3, since TP4 is a worst case testing.

Table 7. Average of total costs - SPT

Group	Method	N	MEAN	% saving (G1)	% saving (mean-1)
G1	no-protection	2,240	211, 927		
G2	mean-1	2,240	65, 535	69.1%	
	upperbd	2,240	80, 368	62.1%	(22.6%)
	mean-2	2,240	75, 883	64.2%	(15.8%)
G3	TP1	2,240	64,966	69.3%	0.9%
	TP2	2,240	66, 034	68.8%	(0.7%)
	TP3	2,240	70, 479	66.7%	(7.5%)
	TP4	2,240	115, 604	45.5%	(76.4%)
$\overline{(G2)}$		6,720	73,928	65.1%	(12.8%)
$\overline{(G3)}$		6,720	67, 159	68.3%	(2.5%)
$\overline{(G2,G3)}$		15,680	70, 543	66.7%	(7.6%)

Further analysis was done on the results from EDD dispatch rule. Table 8 is a summary of the experiment design information, concluding the number of levels tried for each factor. Table 9 and 10 are first order and second order F-tables respectively. Each F value in the table represents the significance of a factor or joint effects of factors to the output of the experiments. A bigger F value shows its greater impact on output when it switches between its levels of values. Each F value is accompanied by a significance index P, signifying the confidence interval of F. e.g., a 0.000* P value means the result is significant at the 0.05 level, which has a confidence interval of 95%. Thus the smaller the P is, the greater confidence we have on the effect of the factor.

From first order F-table, we conclude that the sequence of factor importance in affecting the final cost is (from high to low): TP method, cost type, ratio, due date and mtbf. The F value range is very wide, with that for approach very significant and that for mtbf almost insignificant. This certifies that final cost is most sensitive to the scheduling approach employed than any other environmental parameters.

Joint effects between factors can be seen from the second order F-table. The similar high to low effect can be sequenced as: approach/cost type -> approach/ratio -> ratio/cost type -> approach/due date -> due date/cost type -> ratio/due date -> approach/mtbf -> cost type/mtbf -> ratio/mtbf -> mtbf/due date. Therefore, approach and cost type (or ratio) in the experiments affect the result greatly, while mtbf and due date have a smaller effect no matter what values they are bearing. This further validates the conclusions drawn from the first order F-table. In that table, cost type and ratio are the second and third significant factors apart from approach.

Table 8. Summary of design information

FACTOR	run#	method	due date	mtbf	ratio	ctype	results#
LEVELS	5	8	4	4	4	7	17920
TYPE	random	within	within	within	within	within	data

Table 9. F-table (1st order)

source	method	due date	mtbf	ratio	ctype
F P	27511.298 (0.000***)	269.520 (0.000***)	1.097 (0.388)	936.803 (0.000***)	3013.300 (0.000***)

Table 10. F-table (2nd order interactions)

source	**method**	**due date**	**mtbf**	**ratio**	**ctype**
method P	-				
due date P	348.857 (0.000***)	-			
mtbf P	7.875 (0.000***)	0.582 (0.803)	-		
ratio P	7715.868 (0.000***)	169.017 (0.000***)	1.603 (0.152)	-	
ctype P	20388.414 (0.000***)	306.115 (0.000***)	2.087 (0.015*)	497.188 (0.000***)	-

Figure 15 shows the average total cost generated by each method including the no protection method, where no temporal protection was provided at all. Grand average of all eight methods is labeled g-mean on the graph, while the mean of all without group 1 is labeled m-nobl. Figure 16 - 20 shows the average cost comparison by experiment factors (due date, MTBF, ratio, unit cost type) and eight scheduling methods. Figure 21 outlines the average cost across all methods depending on the levels of each of the four parameter factors.

Comparing the three methods in group two, Mean-1 has the smallest duration extension among the three. Upper-bound can have a smaller or bigger duration protection than Mean-2 depending on the failure data (see EQ6 and EQ10). Apparently Mean-1 dominates the other two methods in our experiments (see figure 16 to 20). Upper-bound method is not preferred compared to the others. It is actually worse than most of the methods except the baseline and TP4. This is because the extension duration of Upperbound, which is $P_{ext1\text{-}lb}$, is too much. i.e., overly protected durations incur more machine idle time when the actual interruption is less than expected.

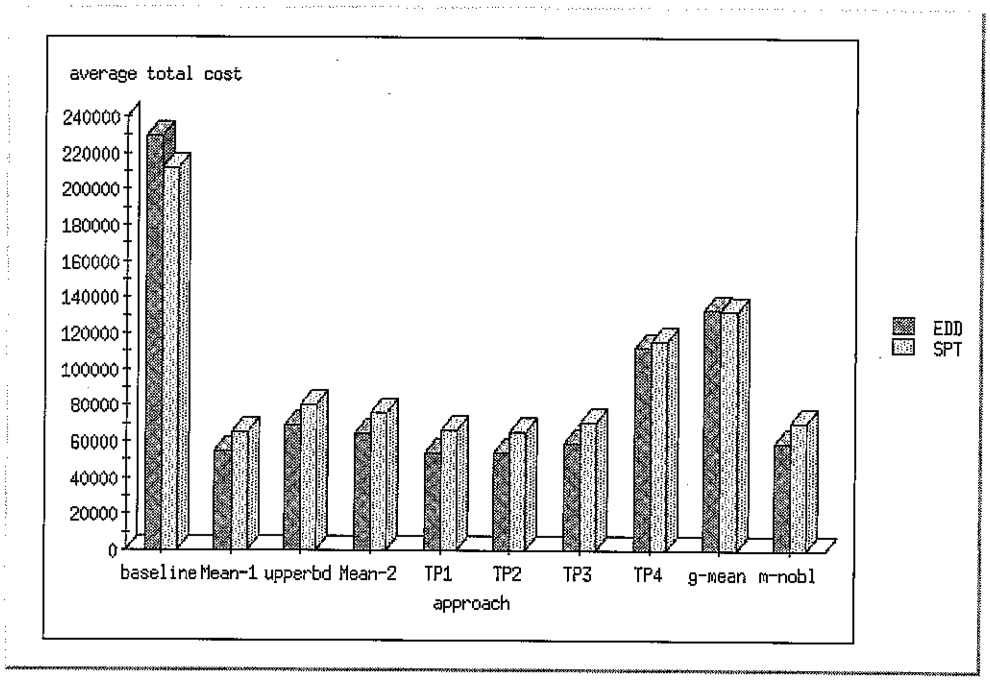


Figure 15. Comparing Total Costs based on Temporal Protection and Scheduling Rules.

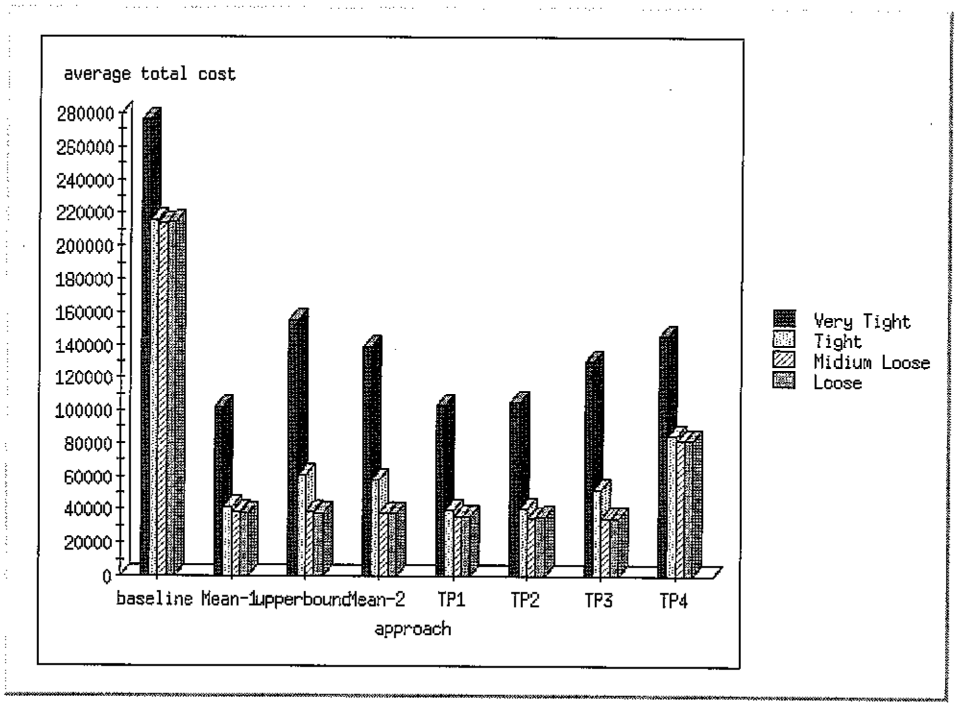


Figure 16. Comparing Total Costs based on Temporal Protection and Due Date Type (EDD).

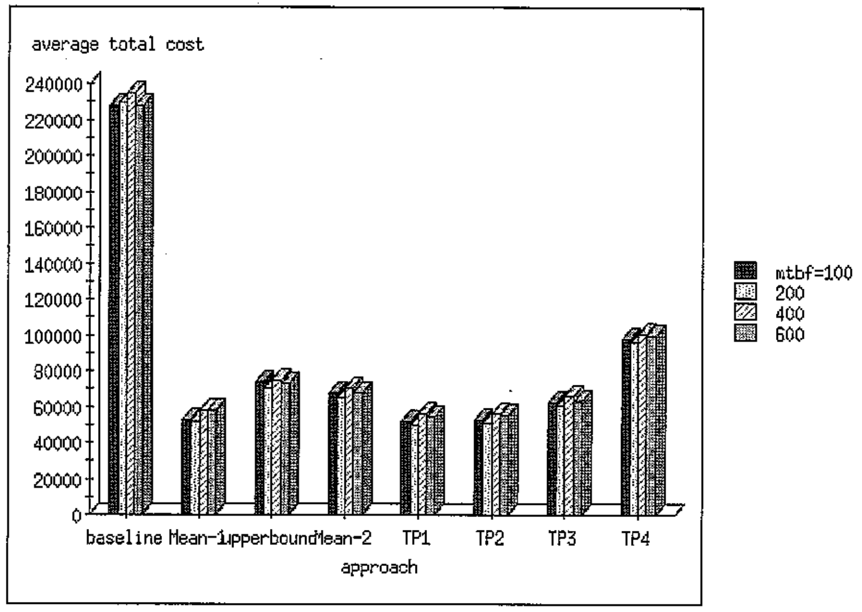


Figure 17. Comparing Total Costs based on Temporal Protection and MTBF (EDD).

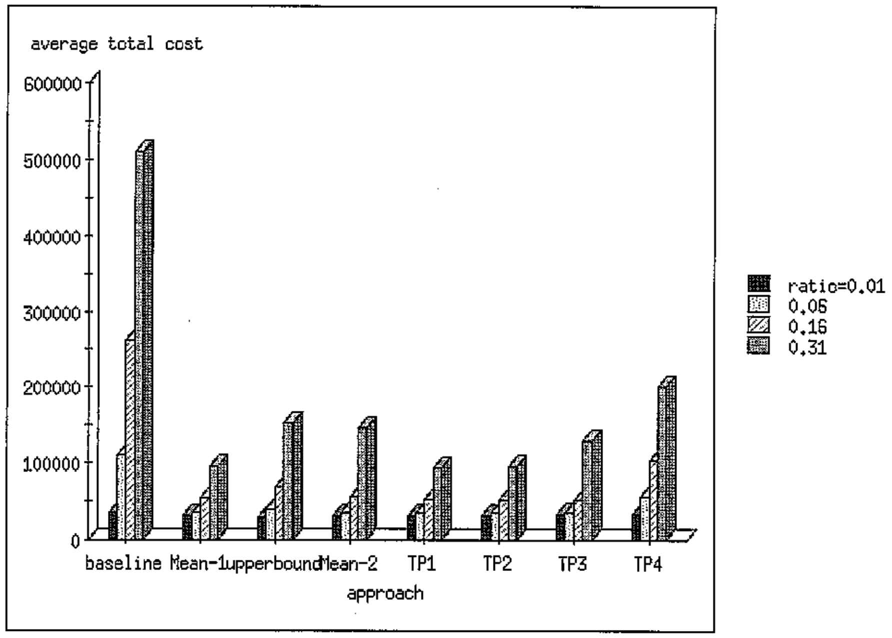


Figure 18. Comparing Total Costs based on Temporal Protection and Ratio (EDD).

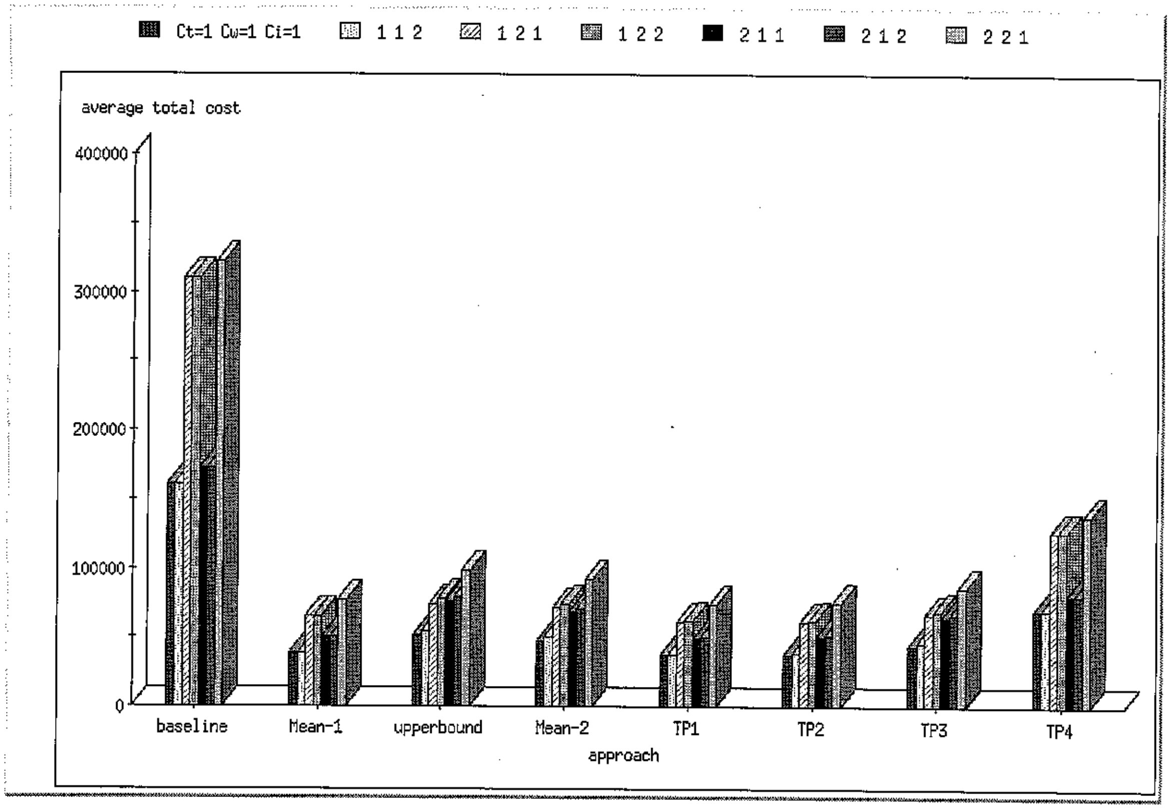


Figure 19. Comparing Total Costs based on Temporal Protection and Cost Types (EDD).

Mean-2 is better than Upperbound but is unsatisfactory comparing to Mean-1. The duration protection of Mean-2 is smaller than that of Upperbound but still too generous. If we take a further look into method TP4, which uses lower bound of Mean-1 as duration protection and results in a much higher total cost, we conclude that Mean-1 is a proper duration protection in our experiments.

Based on the above analysis, we come to the conclusion that Mean-1 is a proper protection for activity duration under machine failure uncertainty. This is also intuitively correct since Mean-1 accounts for machine down time by multiplying the average numbers of failures with the average failure duration. Lower bound of Mean-1 is too restrictive to accommodate actual total down time. Mean-2, as we know, would have worked better if we interpret the machine down time as part of mean time between failures.

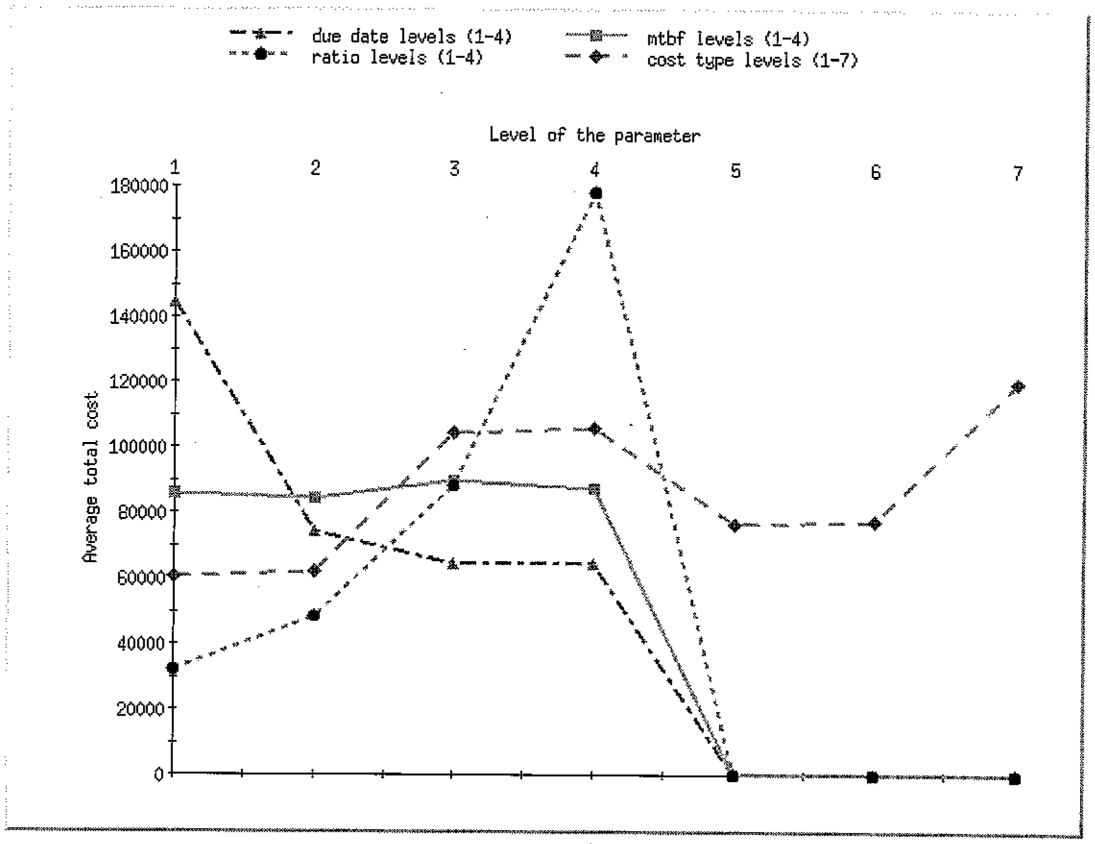


Figure 20. Average for Different Levels of Due Date, MTBF, Ratio and Cost Types.

We now analyze the four temporal protection methods in group three. In Figures 15 through 18, TP1 and TP2 outperform other methods with TP1 slightly better than TP2. Both TP1 and TP2 are extended models of Mean-1. Duration protections are the same as in Mean-1, which has been tested to be a suitable protection for activity allocation. The difference lies in the adjustable release times based on machine uncertainty patterns (F and D). This provides more flexibility in response to deviation in schedule execution and results in a generally lower total cost than Mean-1. The difference between TP1 and TP2 is that TP1 depends on the information of P_{ext1} only, by using mean of P_{ext1} as inner part and upper bound of P_{ext1} as outer part. TP2 includes the information of P_{ext2} for calculating the slack interval (difference between Mean-2 and Mean-1). The fact that TP1 is better than TP2 further validates that P_{ext1} deals with uncertainty information more accurately in our experiments.

TP3 is an extended model of Mean-2. It is understood that when there is no uncertainty on failure data ($D_{lb} = D_{ub} = D_{mean}$ and $F_{lb} = F_{ub} = F_{mean}$), TP3 becomes the same as Mean-2. ($P_{ext2-lb} = P/(1-D_{mean}/F_{mean}) = P_{mean-2}$, $P_{inner} = P_{exp-lower} = (P_{ext2-lb}+2P_{mean-2})/3 = P_{mean-2}$, $P_{lower-slack} = P_{upper-slack} = P_{ext2-lb} - P_{inner} = 0$) As we discussed for Mean-2 method, excessive duration protection causes extra idleness and tardiness costs, which are responsible for TP3's cost increase than TP1 or TP2.

TP4 results in a big increase in total cost (85% more than TP1 and 52% more than TP3). It even performs worse than the methods in group two. It is still far better than the no protection line (57% less) though. TP4 is the least wanted because the inner part of activity reservation does not provide enough protection under uncertainty. In reality, activities take longer than the allocated interval to finish. Subsequent activities wait longer for the machine. Machine idleness is minimized since there is often activities waiting on the floor. But WIP cost is increased. It overrides the other two costs and worsens the performance of TP4 scheduling. In our experiments, WIP cost for 100 activities contributes more to the total cost than that of tardiness or idleness. This implies that the duration protection plays a more important role than release time protection when WIP cost is the primary concern.

When due date is very tight or tight, the cost saving obtained from temporal protection (TP1 and TP2 especially) is more obvious than the situation when due date is loose or medium loose, showing the techniques' capability under very constrained environment. Notice that the cost of Loose due date is almost the same as Medium Loose due date (Figure 16). Therefore, unnecessarily long due date does not lead to low cost.

The frequency of machine breakdown does not have a significant effect on the final result (Figure 17). i.e., whether the activity is frequently interrupted or seldom interrupted, same scheduling approach can be employed. What affects the final cost greatly is the D/F ratio. When failure ratio is small (0.01 or 0.06), the cost differences among all methods are small. The differences increase as the ratio (D/F) increases (to 0.16 or 0.31), showing some methods (TP1, TP2 or Mean-1) better off in dealing longer failure situations.

Figure 18 shows that when the ratio of failure duration to MTBF is very small (i.e., 0.01), all methods generate approximately the same total cost. This is the case when all failures recover quickly and schedule is less affected by the machine breakdown. When ratio is very small, both P_{inner} and P_{outer} are close to the original processing time P. Upper and lower slack intervals are very limited. This is true for all temporal protection methods described in section four. Therefore, if the machine failure can be recovered instantly, there is no need to switch to a TP-type scheduling. This validates the continuity of the temporal protection techniques we proposed.

The total cost used in this experiment is composed of three components: tardiness cost, wip cost and machine idleness cost. From Figure 19, we realize that the crucial cost among the three is wip cost, with tardiness and idleness cost less significant. This explains why fixed bound method Mean-1 is almost as good as TP1 and TP2. Since Mean-1 does not provide release protection, their release times are pushed a bit later. WIP time is therefore relatively low. Although they are more likely to complete beyond due dates, low wip cost guarantees its good performance in our experiments.

Comparing all methods, due to the results, we can generally say that TP1 is the best and Mean-1 is also a good choice. TP1 is appropriate in terms of both duration extension and release slack interval. Release time protection plus duration protection is generally better than duration protection alone. What stands true is that all methods with temporal protection (group 2 and 3) offer far better results than the case with no protection at all. Temporal protection is definitely encouraged to generate robust schedules in a failure-prone environment.

4.4.1 Conclusion from this initial experiment

In domains containing high degrees of uncertainty, whether due to machine failure, resource unavailability or poor performance of tasks, it is necessary to increase the robustness of predictive schedules. In this section, we have investigated a method called Temporal Protection for constructing robust schedules. The method mitigates machine failure uncertainty through the manipulation of a activity's release time and extended duration. The concept of an earlier resource release time than activity start time is introduced as a means of achieving temporal flexibility. By analyzing failure data, the protection technique increases an activity's duration and/or changes its releases time so that its availability temporally overlaps with other activities. The experiment was designed to compare the schedule generated with no protection, schedules generated with duration protection, and schedules generated with both duration protection and release time protection. The results show that TP1 is the best temporal protection model among those tested. And protecting both duration and release time is better than extending duration alone. The method works well for schedules that is created by both EDD and SPT dispatching rules.

TP1 is the main technique we are going to incorporate into the scheduling tool ODO:TNG. More experiments will be implemented on multiple machine job shop scheduling problems in chapter 6.

Chapter 5 Incorporating Temporal Protection into ODO:TNG

This chapter first gives an overview of the ODO:TNG scheduling tool. It then presents the nature of the scheduling problems we intend to solve. Resource failure representation is added into the constraint graph representation. Implementation of uncertainty representation and problem solving in ODO:TNG is discussed.

5.1 Overview of ODO:TNG

5.1.1 Introduction

Research in constraint based scheduling has focused on graph-based constraint satisfaction and optimization techniques [Sadeh 91] [Zweben 94] [Davis 94]. In this approach, a problem is represented by a constraint graph, where the nodes are the variables of tasks and resources, and the arcs are constraints among the variables. Solving a problem amounts to assigning values to all variables such that all constraints are satisfied.

There are two common methods for solving scheduling problems represented in constraint graph. The constructive method [Sadeh 91] [Fox 87] starts with an empty schedule and assigns a value to a variable only if it is consistent with all previous assignments; if the current set of assignments cannot lead to a feasible solution, then the method backtracks and tries again. The iterative method [Zweben 94] [Minton 92] starts with values assigned to all variables and repeatedly modifies those values until all constraints are satisfied. Both constructive and iterative methods continually modify the current schedule in a search to find a solution as quickly as possible.

The successful search heuristic usually exploits structural properties of the given problem. ODO:TNG uses the term *textures* to describe these properties when the problem is represented in a constraint model.

ODO:TNG is a constraint-based scheduling architecture that was built to be a platform for exploring many issues. It employs both constructive and iterative search methods, and bases heuristic decisions on problem property measures (textures). With the architecture's command language a user specifies a problem to be solved and the search parameters used in solving the problem. ODO:TNG can be used to test a variety of constraint based scheduling methodologies, as well as to study the relationship between problem textures and efficient search heuristics for both generative and iterative scheduling methods. However, there is no reason why ODO:TNG can not be used to model and solve real world scheduling problems.

5.1.2 Constraint Model of Scheduling

In ODO:TNG's constraint model, problems are represented by a collection of objects, variables, and constraints. The objects serve as placeholders for variables and constraints, and may be used to store measured texture information. Figure 21 presents a hierarchy adequate to represent simple job-shop scheduling problems, along with a contention graph for a simple problem instance. As noted in the hierarchy, objects are represented as boxes, variables as hollow circles, and constraints as filled circles with lines drawn to the relevant constrained variables.The problem being addressed is job-shop scheduling with due dates. Though there exist constraint representations for more complex constraints (see for example [Zweben 93]), initially attention is restricted to precedence and resource constraints. The scheduling problem in Figure 21 describes:

- Variables: start and end times of all activities, and the resource assignments for activities (if any).
- Temporal constraints: for each activity, start-time+duration=end-time,

  start-time(Task3) >= end-time(Task1)

  start-time(Task3) >= end-time(Task2)
- Resource constraints: the assigned resource must be allocated to the activity from its start-time to its end-time.

5.1.3 Textures

A texture is a property of a constraint graph. These texture estimates are what the heuristics are a function of. For example, in backtracking search, the next schedule modification is chosen that will least likely cause backtracking to occur. In the constraint model, this modification principle can be summarized as follows: find the *most constrained* variable, and assign it a value that *least constrains* all later assignments. The notion of *most constrained* variable and *least constraining* value relate to the concepts of *variable tightness* and *variable/value goodness* textures introduced by [Fox 89] . In MicroBOSS [Sadeh 91] , for example, the most constrained variable is the activity that relies upon the most contended resource/time reservation, and the least constraining value is that reservation estimated as having the highest probability not to conflict with later activity-resource/time assignments.

In iterative search, the next modification is chosen that will hopefully reduce the number of violated constraints by the greatest amount. One approach, a variant of MIN-CONFLICTS [Minton 92] , selects the activity participating in the most number of violations and moves it to the time that would result in a schedule with the fewest number of overall violations.

ODO:TNG implements a variety of heuristics that are functions of these textures for both generative and iterative cases. One important research component within ODO:TNG is to test heuristics' performance on problems with different textures.

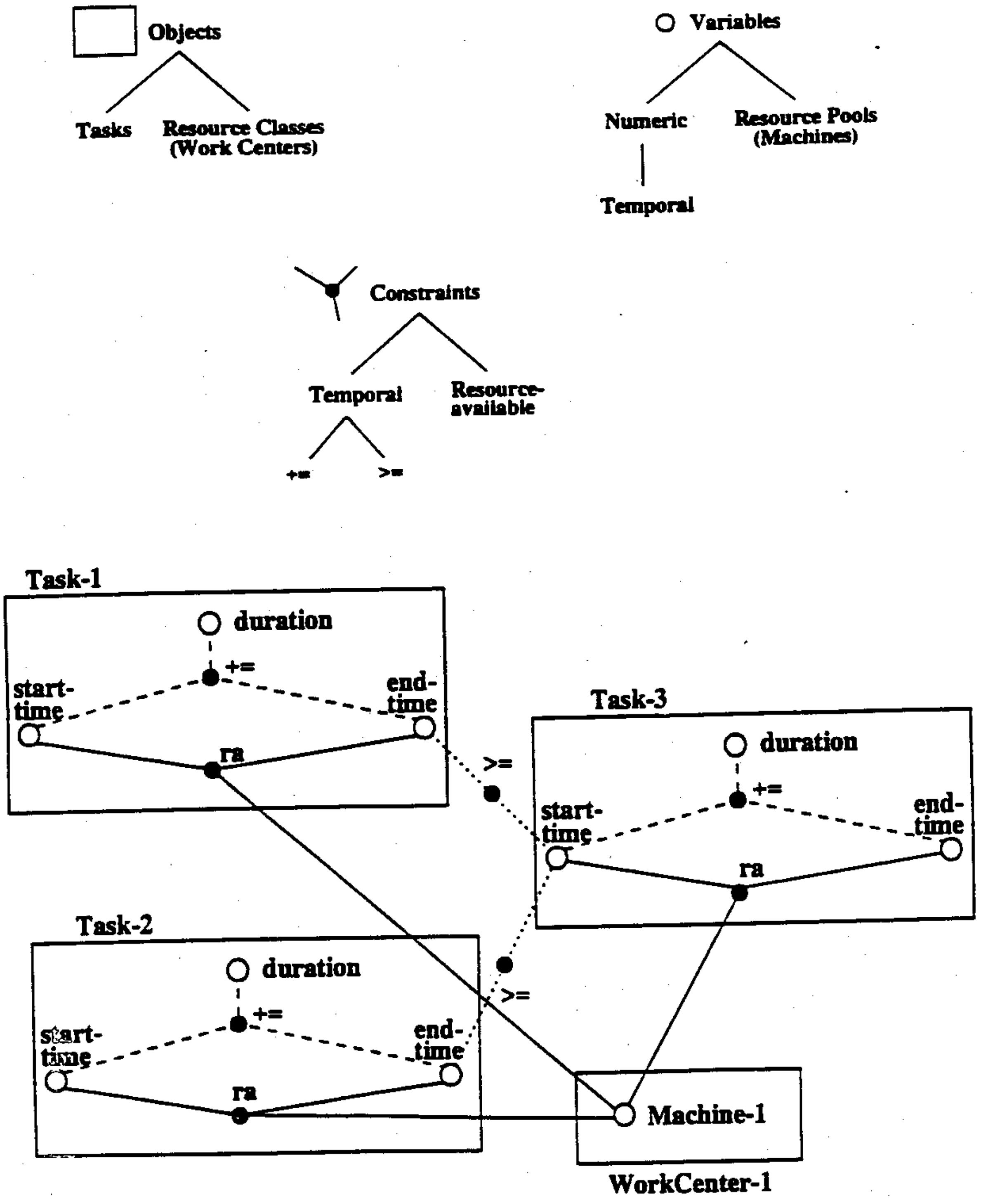


Figure 21. ODO:TNG Constraint Model of Scheduling

5.1.4 Search Techniques

Within the constraint based framework, a scheduler starts with a state-transition model where a transition is an assertion or retraction of a "commitment" (assign a value to a variable, etc.). An overall strategy on how transitions are made is constructed as a "policy" in ODO:TNG. Problem solving is a process of commitments assertion/retraction and is guided by the user defined policy. A policy dictates how variable/value are selected, how the assertion is propagated, how resulting state is evaluated, how to backtrack if the new state is rejected, and etc. A policy is followed until some termination condition occurs, at which point search can terminate or a new policy can begin.

Search ends when the last policy specified have reached its termination criteria. A full or partial schedule (e.g., some activities do not have a valid start time assigned) is delivered. If constraint posting is used as commitment type rather than variable assignment, the resulting schedule shows intervals for all start/end time variables instead of a single start/end time.

Further extensions of the ODO:TNG include strengthening the policy structure so that more than one type of commitments (start time assignment and resource assignment) can be made in one state. More discussion about ODO:TNG design and implementation can be found in [Chris 95].

5.1.5 Built-in Language

A well-defined, extensible PrOblem Description Language (PODL) is devised as a user interface to ODO:TNG's solving mechanism. More examples of problem definition using PODL can be found in chapter six.

5.2 Problem Description

The scheduling problem being addressed here is a variation of simple job shop scheduling problem as described in [Davis 94]. The main difference is that random resource failure is allowed in our problem. This relaxation of assumption greatly increased the complexity of problem solving. Scheduling amounts to a similar constraint satisfaction problem where resource assignment and activity start time assignment are to be made so that all temporal/resource constraints are satisfied. On the other hand, the activity network is uncertain in terms of activity duration and resource availability. More specifically, the problem is composed of:

- a set of *resources*, each with a positive integer *capacity*. Some or all resources are subject to failures, whose occurrence can be stochastically modeled by two random variables: mean time between failure and failure duration.
- a set of jobs; each job with a positive *due date* and consists of a set of ordered *activities*; no multiple process plans exist; each activity has positive *duration*; each activity may require single or conjunctive resources, among which failure prone resources (usually machines) are called critical resources and non-failure prone (usually materials) are called noncritical resources.
- a set of *criteria*; a challenge exists which is to find a systematic way of proving one schedule's robustness over another. Cost of schedule execution is a tangible standard for this request. Due to the dynamic feature of the resource environment, we need a simulator which can repeatedly execute a predictive schedule under a randomly generated failure environment. The success of a schedule depends on the resulting execution cost incurred (e.g., work-in-process cost, tardiness cost, etc.).

The impacts of resource failure uncertainty on scheduling is the focus of this work. Resource uncertainty refers to resource's failure behavior from time to time, though no knowledge of exact failure time can be obtained in advance. We refer to resource failure uncertainty that can be probabilistically modeled by mean time between failure and failure duration. "Unknown" events such as failures due to a power off are not considered.

Resource uncertainty presents significant challenge to scheduling due to its direct impact on activities scheduled on failed resource and its indirect impact on activities which are temporally related to the activities being affected in the first place. What is more, more than one resource on the shop floor can be failure prone, each with different degrees of failures. Problem representation of ODO:TNG allows an activity to require both conjunctive and disjunctive resources, among which all or some may contain failure uncertainty.

Remaining of this chapter will discuss the original activity constraint representation (introduced by [Davis 94]) and our extension to model scheduling under resource failure uncertainty. Schedule generating incorporates temporal protection technique TP1 (discussed in chapter 4) to prepare the predictive schedule for potential resource failures. Simulation mechanism is designed for testing robustness of generated schedules under a random resource failure environment.

5.3 Activity Constraint Graph in ODO:TNG

In our description of the job-shop problem, jobs are composed of activities, and activities are the entities which must be assigned resource and valid execution times in a satisfying schedule.

All activities have *durations* associated with them; a duration specifies the relation between the activities's start and end event time points. Expressing the relation between an activities's start-time, end-time, and duration is done in the form of a temporal constraint between three variables:

$st(T) + dur(T) = et(T)$

Figure 22 illustrates the basic activity object composed of three variables and a temporal += constraint. Variables are drawn as empty nodes. Arcs and solid node represent activity internal constraint between the variables.

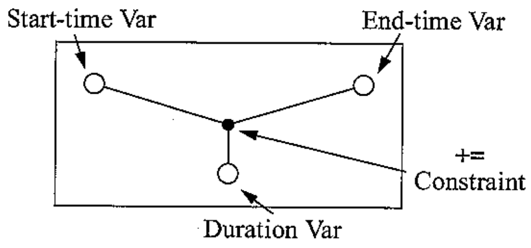


Figure 22. Basic Activity

The activity may have temporal constraints (any of Allen's [Allen 84] 13 possible temporal relationships such as *before*, *after* and *overlap*) with other activities. A *resource request variable* (rrv) is created for each of its resource requirements. The rrv will be assigned a resource object as its value when resource commitment is asserted. A *resource constraint* ties the rrv to the start and end time variables of the activity. The constraint is satisfied only if the resource object assigned to the rrv has the requested amount of capacity available from the start to the end time of the activity.

5.4 Representing Activity Uncertainty

5.4.1 Random duration

The first extension to the problem representation is to represent the random characteristic of the activity duration. Activity duration should be able to be specified as a random number when fixed value of duration is not known. A new description line called **random-duration** is added into PODL problem description language [Chris 95] for this purpose. For instance, if an activity's duration follows uniform distribution of U[10, 15], then the representation is written in PODL problem description language as:

```
(activity
   :name act1
   :random-duration (UNI 10 15))
```

Other commonly applied distributions include exponential distribution, normal distribution, triangle distribution, and etc. Examples of representation are respectively:

```
(activity
   :name act2
   :random-duration (EXP 0.5))

(activity
   :name act3
   :random-duration (NORM 10 0.1))
```

A random number generator is linked to these representations. Before the problem solving starts, a random number following the specific distribution is generated and used as the real activity duration.

5.4.2 Unknown Duration

Under some circumstances, the duration of an activity is simply unknown. e.g., if the decision of buying or producing a part has not been made, the duration of procuring that part is unknown. The ability of not committing to any description should be allowed. In Podl description, it can be specified as following.

```
(activity
   :name unknownDuration
   :random-duration unknown)
```

The actual processing is that min-duration is set to schedule min-time, and max-duration is set to schedule max-time, i.e., if we do not know how long it takes to execute an activity, then it can take as short as the minimum length, or as long as the maximum length of the scheduling. Depending on the other activities being scheduled, forward and backward temporal propagations modify the min-duration and max-duration of the unknown activity. The final range of duration determines the flexibility that the activity duration can take.

5.4.3 Distinguish resource release time from activity start time

As first introduced in chapter 3, an activity often requires multiple resources, of which failure machine is treated as critical. While we can not predict the availability of the failure machine at a certain time, other resources are released at an arranged time. The incentive to distinguish between resource release time and activity start time is to take the opportunity when the uncertain machine is released earlier than planned. Temporal protection provides the flexibility that release time for non critical resources is earlier than the planned start time of the activity. How early it can be released depends on the machine uncertainty and the activity being protected. Conventional scheduling releases the uncertain resources at the same time the activity is scheduled to start. Our method takes advantage of machine uncertainty, i.e., if the previous activity finishes earlier because of less actual machine breakdown than expected, the current one can start earlier than its scheduled start time, as early as the resource release time.

Figure 23 shows the modified constraint based activity representation. Implementation code is also expanded to denote the notion of release time. Release time is related to start time by another += constraint, difference being the early release interval. Note that the protected duration and early release interval depend on both the activity and the resource being assigned. Failure information is stored in individual resource object. Accessing assigned resource is through the activity's list of Resource Request Variable.

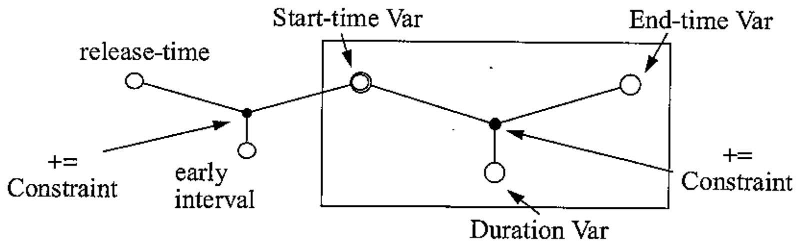


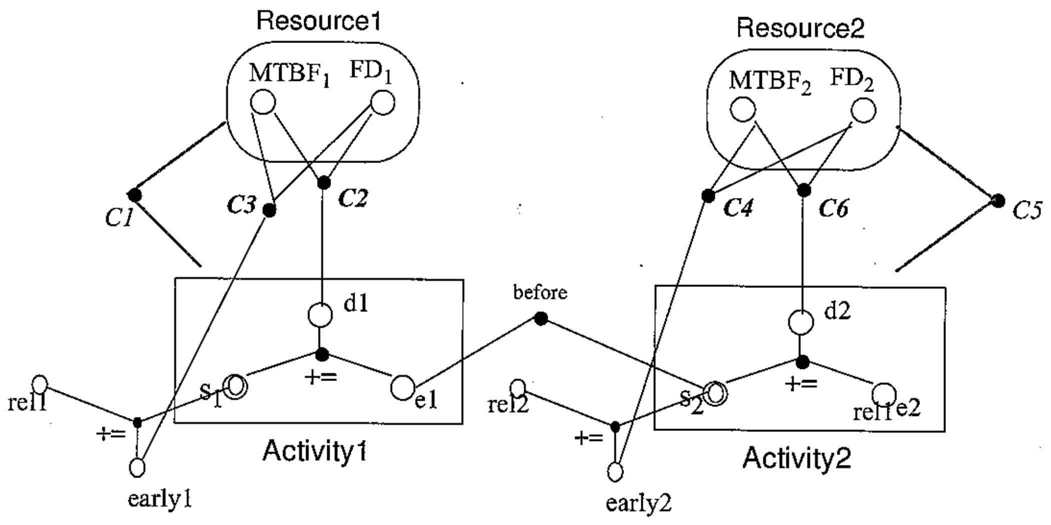
Figure 23. Activity with release time

```
Class Activity
{
   .
private:
   IntervalVariable *_start, *_dur, *_end;
   int _releaseTime; // _release = min-start - _earlyInterval
   int _earlyInterval;
   .
)
```

5.5 Representing Resource Uncertainty

5.5.1 In the Activity Network

Machine failure is the only uncertainty addressed in this work. Figure 24 proposes an extension to simple activity network to represent machine failure information and its impact on the activities that work on the resource. Two failure data are important: one is the Mean Time Between Failure (MTBF), which is modeled as a random distribution such as uniform, triangle, exponential and normal. The other data is the Failure Duration (FD), which can also be modeled as random distribution. The selection of distribution bases upon the machine characteristics and the knowledge of its performance. The minimum, maximum and mean values of MTBF and FD are retrieved from the description, since temporal protection only uses the mean value and two bounds value of a distribution in calculating the protection amount. Take uniform distribution, for example, a MTBF described as U(10, 20) gives $MTBF_{mean} = 15$, $MTBF_{mean} = 10$, and $MTBF_{mean} = 20$.



s_i: start time variable for activity i.
e_i: end time variable for activity i.
d_i: duration variable for activity i.

MTBFi: Mean Time Between Failure for machine i.
FDi: Failure Duration for machine i.

$$C2\text{: } d_1 = d_1 + d_1 / MTBF_{mean\text{-}1} * FD_{mean\text{-}1}$$
$$C6\text{: } d2 = d_2 + d_2 / MTBF_{mean\text{-}2} * FD_{mean\text{-}2}$$
$$C3\text{: } early1 = (d_1 / MTBF_{min\text{-}1} * FD_{max\text{-}1} - d_1 / MTBF_{mean\text{-}1} * FD_{mean\text{-}1}) / 2$$
$$C4\text{: } early2 = {}_2 / MTBF_{min\text{-}2} * FD_{max\text{-}2} - d_2 / MTBF_{mean\text{-}2} * FD_{mean\text{-}2}) / 2$$

Figure 24. Representing Resource Uncertainty in Constraint Graph

In the above graph:

- C1 and C5 are normal "use-resource " constraints.
- C2 and C6 are constraints placed among the MTBF variable, the FD variable and the activity duration variable, representing duration extension based on the reliability of machines. The definition of C3 and C4 are based on temporal protection technique called TP1 [Gao 95] .
- If uncertainty on the machine failure changes (e.g., a machine gets older and tends to fail more frequently), the distributions of MTBF and FD are updated, causing all activities on it to update their values of protection.

5.5.2 PODL representation of machine uncertainty

Machine failure information is to be input by the user in Podl problem definition. The following example describes one resource role class of the mechanism role type. Machine1 plays the role of Milling with a simple capacity of 1. Machine1 fails every 20 to 30 time units, with each failure lasting 2 to 5 time units. It also states that temporal protection TP1 is used in building the schedule so that activities on Machine1 will be protected accordingly.

```
(role-class :name Milling :role-type Mechanism)

(object :name Machine1
         :has-role (role :name Milling :sim-capacity 1)
         :mtbf (UNI 20 30)
         :fail-duration (UNI 2 5)
         :protected TP1)
```

This representation allows the user to specify the mean time between failure (MTBF) and failure duration (FD) for a resource if he or she knows that information before scheduling. If this information is not specified, the default assumption is that the machine is not subject to failures. Bounds on MTBF and FD (minimum and maximum) are extracted from the Podl description and will be used in calculating temporal protection to an activity scheduled on this resource object.

Following is a simple but complete (excluding policy file) example of problem declaration written in PODL grammar.

```
(schedule :name sched1
            :min-time 0
            :max-time 100)
(role-class :name Milling :role-type Mechanism)
(role-class :name Lathe :role-type Mechanism)
(role-class :name PowerDrill :role-type Mechanism)

(object :name Machine1
         :has-role (role :name Milling :sim-capacity 1)
         :mtbf (uniform 20 30) :fail-duration (uniform 10 15))
(object :name Machine2
         :has-role (role :name Milling :sim-capacity 2))
(object :name Machine3
         :has-role (role :name Lathe :sim-capacity 1)
         :mtbf (uniform 20 25) :fail-duration (uniform 8 10))
(object :name Machine4
         :has-role (role :name PowerDrill :sim-capacity 1))

(activity :name a1
         :random-duration (uniform 10 20)
         :uses (object :object-name Machine1 :role-name Milling
         :amount 1))
(activity :name a2 :min-duration 10 :max-duration 10
         :temporal-relation (after :activity a1)
         :uses (object :object-name Machine1 :role-name Milling
         :amount 1))
(activity :name a3 :min-duration 10 :max-duration 10
        :temporal-relation (after :activity a2)
        :uses (object :object-name Machine3 :role-name Lathe
        :amount 1))
(activity :name a4 :min-duration 10 :max-duration 10
        :temporal-relation (after :activity a3)
        :uses (object :object-name Machine4 :role-name PowerDrill
        :amount 1))
```

5.6 Calculating Temporal Protection in ODO:TNG

As mentioned earlier in this chapter, ODO:TNG allows an activity to require both conjunctive and disjunctive resources (Figure 25), among which all or some may contain uncertainty.

Protected duration and early release interval can be calculated like that in Figure 24 if the activity requires a single resource. When conjunctive and disjunctive resources are involved, aggregation technique is needed to estimate the amount of protection needed on the activity.

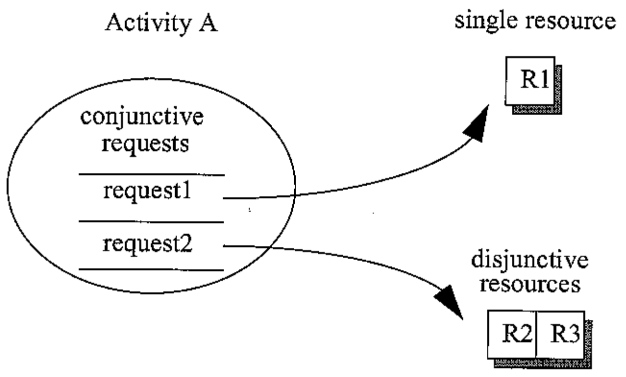


Figure 25. Activity can require both conjunctive and disjunctive resources

For example, assume R1, R2 and R3 in Figure 25 are all subject to failures. Based on request 1, activity A is protected by extended duration d1 and release interval r1 according to figure 24. Request 2 generates protection d2, r2 or d3, r3 depending on whether R2 or R3 is assigned to this request. The total protection on activity A depends on how the scheduler is going to aggregate from request 1 and 2. The following sections discuss the aggregation in more detail.

5.6.1 Single resource object

If there is only one resource object R that the activity is requiring, and R contains failure information ($MTBF^R_{mean}$, $MTBF^R_{min}$, $MTBF^R_{max}$, FD^R_{mean} FD^R_{min}, FD^R_{max} are non zero), we calculate the protected duration d' and early release interval rel using TP1 method from chapter four. Equations 14 and 15 calculate d' and rel based on failure information of R and activity's original uninterrupted duration d.

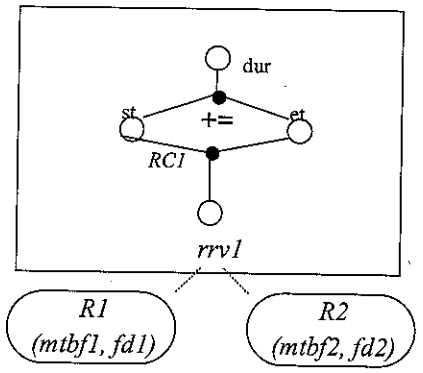
$$d' = d + \frac{d}{MTBF^R_{mean}} \cdot FD^R_{mean} \qquad \textbf{(EQ 14)}$$

$$rel = \left(\frac{d}{MTBF^R_{min}} \cdot FD^R_{max} - \frac{d}{MTBF^R_{mean}} \cdot FD^R_{mean} \right) \quad \textbf{(EQ 15)}$$

Continuity exists, i.e., if R is not subjected to failure, $MTBF^R_{mean} = MTBF^R_{max} = MTBF^R_{min} = 0$, $FD^R_{mean} = FD^R_{max} = FD^R_{min} = 0$, then $d' = d$ and $rel = 0$.

5.6.2 Disjunctive resources

A resource request variable (rrv) is created for each resource requirement. The domain of the variable is a list of possible resource objects that can satisfy the request (Figure 26).The rrv will be assigned a resource object as its value when resource commitment is asserted. Resource constraint is instantiated among the rrv and the start and end time variables of the activity. In figure 26, if MTBF and FD for R1 and R2 are different, different protection d' and rel will apply depending on which resource the activity is assigned to.



rrv1: resource request variable, possible domain values are resource object R1 and R2.

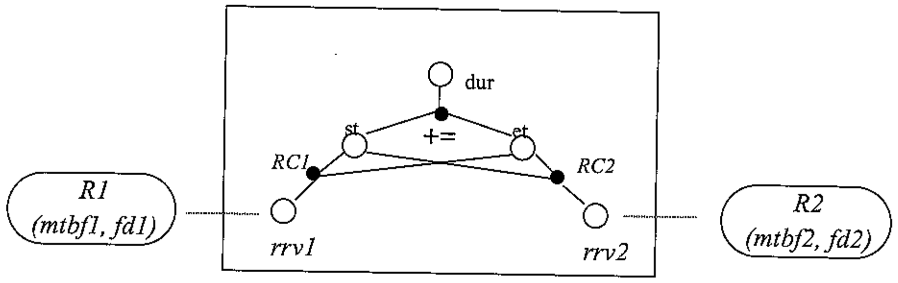
RC1: resource constraint among rrv1, start time and end time variables, instantiated with the assigned resource after resource commitment.

Figure 26. Alternative resources for a resource request

On the other hand, we do need an estimate of the protection (duration extension primarily) in building initial arc-consistency before the resource assignment. We employ a worst case policy which chooses the maximum protection among the alternatives, being most conservative about the uncertainty on this activity.

5.6.3 Conjunctive resources

Considering the case where there are two or more rrvs on an activity such as shown in Figure 27. If both R1 and R2 are subject to machine failures, combined protection on the activity is an aggregation of both. We use the sum of both protections because activity is prolonged whenever either of the machine is down (Figure 28). This estimate is more than reality if there are overlapping breakdowns between R1 and R2. Therefore, it is a worst case estimate for the longest possible breakdowns.



rrv1: resource request variable, possible domain value is resource object R1.
rrv2: resource request variable, possible domain value is resource object R2.
RC1: resource constraint among rrv1, start time and end time variables.
RC2: resource constraint among rrv2, start time and end time variables.

Figure 27. Multiple resource requests of an activity

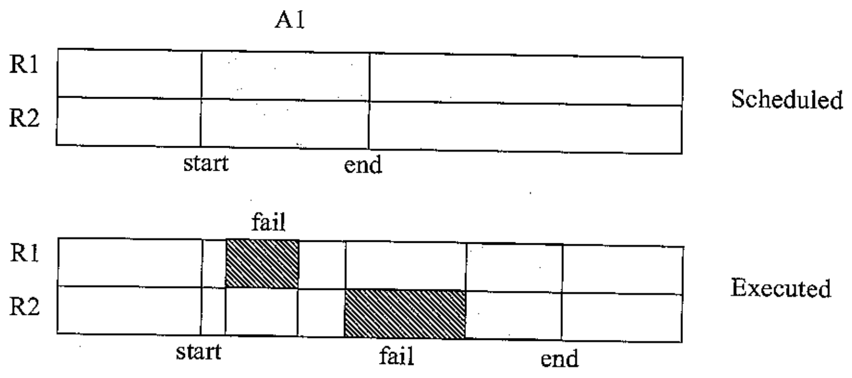


Figure 28. Activity prolonged by failures on both resources

5.7 Scheduling with Temporal Protection in ODO:TNG

Incorporating Temporal Protection into ODO:TNG introduces changes in many aspects of the problem solving. The following sections highlight the changes we made to be able to manage uncertainty during scheduling.

5.7.1 Building Activity Network

Interpreting problem file written in Podl and constructing the whole activity network is the first step of scheduling. Calculation of temporal protection is accomplished at this step. For all activities which are subjected to interruptions due to machine breakdown, their extended durations and early release intervals are computed. The activity network is constructed, using the protected information instead of the original description in the Podl file. Specifically, activity's duration variable and internal += constraint are instantiated using values calculated by temporal protection. We believe that the new network reflects the dynamic nature of the problem, so that a schedule generated from it will perform well under a failure environment.

5.7.2 Resource Assignment

Resource selection is determined by the resource assignment policy and is not affected by temporal protection directly. However, choice of resources does affect the temporal protection applied to the activities being scheduled. If there are disjunctive alternative resources, an estimate of protections is used by initial arc consistency (we used worst case estimate of summation). This estimate may be different from the protection by the resource really assigned. As a result, activity's protected duration and release interval need to be adjusted. We now have a slightly different activity network than before. Network consistency must be reachieved by calling forward and backward propagation from the changed point.

5.7.3 Adding Release Time Assignment

The notion of release time was introduced into ODO:TNG by this work. The value of release time depends solely on start time and the early release interval. Remember that release interval is part of temporal protection calculation and is finalized before start time commitment. Therefore, we simply assign the release time to start time minus release interval after a start time commitment has been asserted. Again, continuity exists. For activities which do not require failure resources, zero release interval is defaulted. Release time will be the same as the start time.

5.7.4 Modify Resource History

Resource history describes the quantity of a resource over time and is updated when an activity is assigned a start time. The critical machine is allocated for the period of the activity (start to end time). The non-critical resources (materials, assembly parts, staff, etc.) are ready early at the release time. Figure 29 shows the difference in resource history modification. It assumes unit capacity for resource and unit consumption for activity. According to ODO:TNG's definition, a reusable resource is freed at the end of the activity, but consumable resource's capacity is consumed to the end of the scheduling.

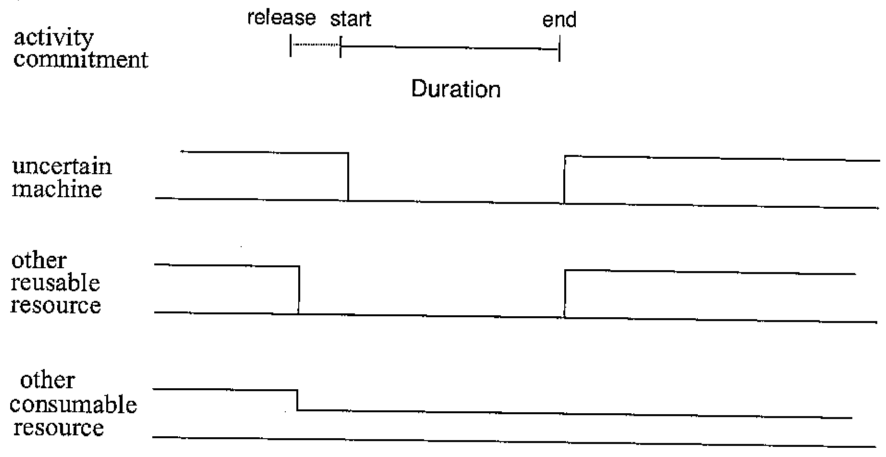


Figure 29. History modification under machine uncertainty

The implementation of this difference is within resource constraint, since resource history is modified when a specific resource constraint is activated. Partial implementation code follows:

```
void ResourceConstraint::activate()
{
// _startTime depends on the criticality of the resource.
if ( (act->getReleaseProtectionGap() != 0) &&
       (_requiredResource->getMTBFmin() == 0) )
   _startTime = act->getStartMin() - act->getReleaseProtectionGap();
else _startTime = act->getStartMin();

_endTime = act->getEndMax();
_requiredRole->changeHistory(_startTime, _endTime, _amountRequired);
}
```

5.7.5 Temporal Propagation

Figure 30 lists three possible temporal propagation scenarios. It is obvious that temporal relationship between activities does not change because of the introduction of release time. Temporal propagation can be carried out the same way as before.

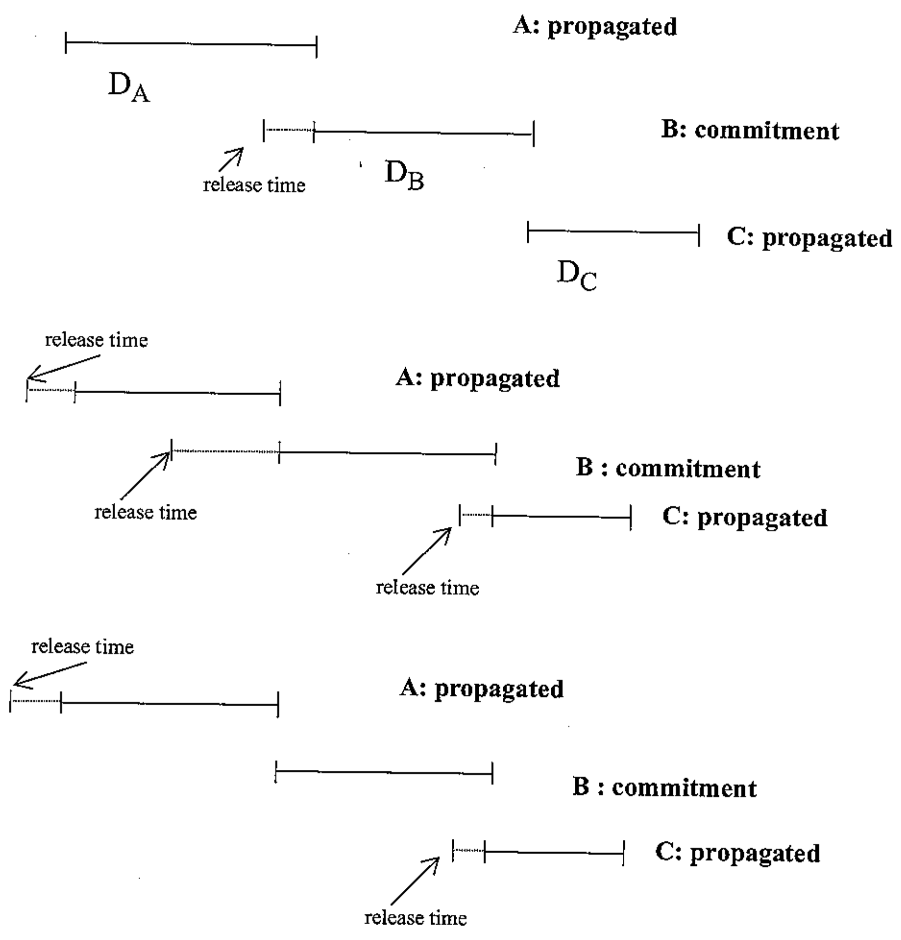


Figure 30. Activities (on uncertain machine) with precedence constraint

5.7.6 Resource Propagation

Unit resource propagation is accomplished after a start time commitment. If an activity is scheduled on a resource for the interval of [low, high], other activities assigned on the same resource should remove the appropriate possible values from the domain of it's start time. To avoid the violation, the unit resource propagation cuts the interval

[low-duration+1, high-1]

from the domain of start time variables of other activities requiring the same resource as the committed activity.

Like the change to history modification, critical and non critical resources have to be distinguished in resource propagation. Propagation on the uncertain resource(s) is no different, since the machine is allocated the same duration as the activity. But the noncritical resource(s) are allocated with an extra release interval. Thus the unit propagation should remove an interval of

[low-duration-protectionGap+1, high-1]

from the domains of start time variables of other activities on the same resource. The implementation is also taken care of by the resource constraint. The proper selection of start time guarantees the corresponding interval is used in resource propagation.

5.8 Simulation of Schedule Execution

This is an addition to the ODO:TNG scheduling tool. Original problem solving stops at the trace and output block in Figure 31. We add two more option steps to simulate the schedule just generated and produce a report on the simulation result. The choice of doing simulation after scheduling is specified by a simple command at the end of the Podl policy file. Therefore, simulation starts with the final state after scheduling.

```
(start-scheduling :policy construct1)
(simulate)
```

Simulation can be required whether the generated schedule was temporally protected or not. (It will simulate the no failure case as well, though execution will be the same as scheduled.) Comparison experiments can be designed (see next chapter) to validate schedule robustness provided by temporal protection.

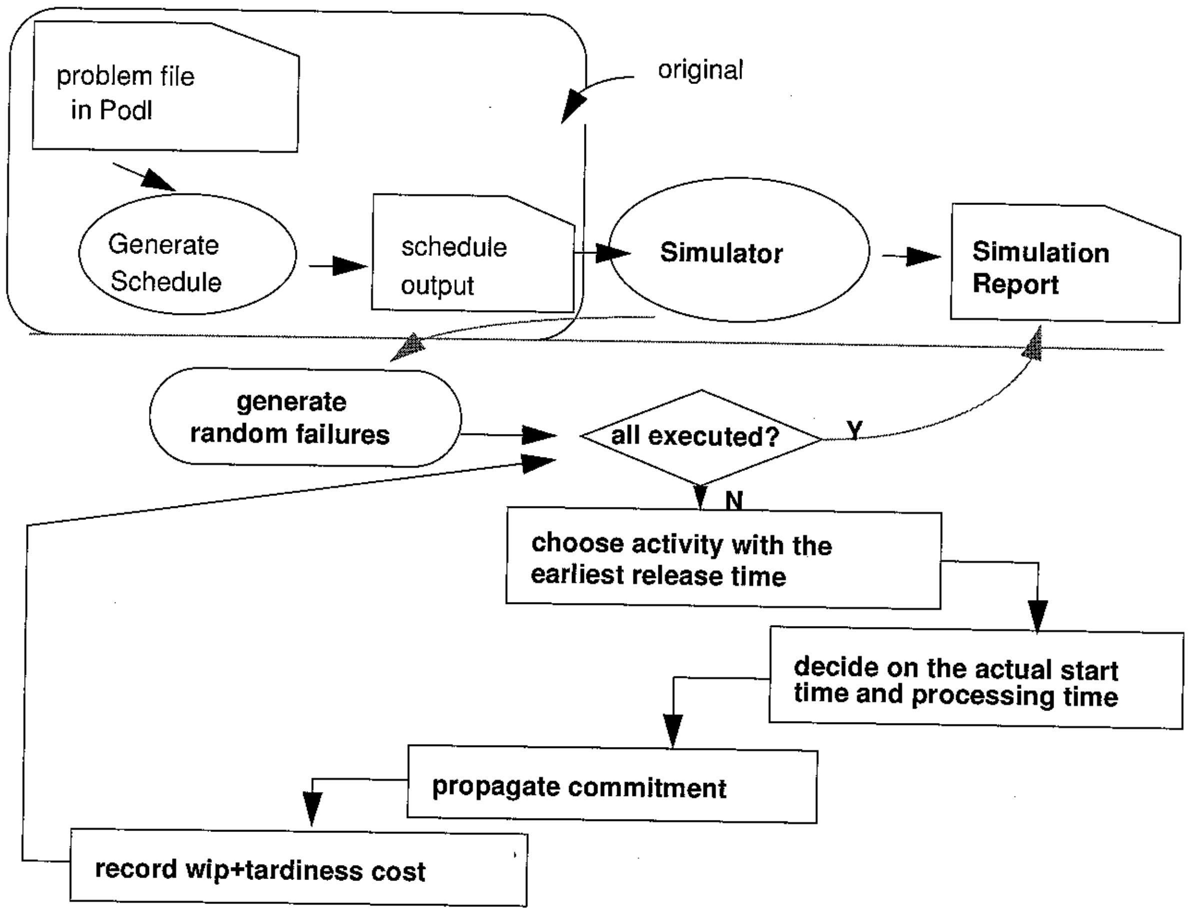


Figure 31. Problem Solving and Simulation Process

The simulation algorithm is detailed in Figure 32. In chapter four, we used a simple dispatching simulation for single machine case. But it does not contain the temporal and resource propagation which we still need in deciding real start time. The simulation presented here uses the same constraint based mechanism as in the scheduling generating process, so that all original constraints are still respected. A queue of breakdowns on each uncertain resource will be generated according to failure distributions. The scheduled activities are then executed in that environment. Dispatching flexibility include moving the start time earlier if the opportunity arises. A report is generated on schedule implementation cost and execution deviation from planned.

Let S be the set of scheduled activities , d_i be the original uninterrupted duration, and rt_i be the planned release time of activity i. Actual start time and duration for an activity is denoted as st'_i and d'_i. Simulate the schedule as follows:

1. Store the planned start time and end time for all scheduled activities.

2. Generate a list of failure intervals for each critical resource according to the description of failure distribution.

3. Reset the duration variables of protected activities to original uninterrupted ones. Call initial arc consistency on the new network.

4. Sort all activities according to the earliest release time. For each activity:

   4.1. Set real start time st'_i depending on planned release time rt_i, previous activity finish time, current start time, and the time of next failure.

   4.2. Check if any failure intervals on the resource overlaps the activity interval $[st'_i, st'_i + d_i]$. Including any interruption into the actual activity duration d'_i .

   4.3. If $d'_i \neq d_i$, change the duration variable for this activity. Reachieve network consistency.

   4.4. Record wip cost, idleness cost and tardiness cost accordingly.

5. Generate statistic report on the total of wip, tardiness and idleness cost. Also reported are schedule deviation such as scheduled and actual makespans.

Figure 32. Simulation of Schedule Execution in ODO:TNG

The next chapter will discuss the experiments and results that are accomplished using this simulation mechanism implemented in ODO:TNG.

Chapter 6 Experimenting Temporal Protection with ODO:TNG

Experiment scenarios were designed to test the temporal protection mechanism in ODO:TNG. In each experiment, a temporally protected schedule was generated by ODO:TNG using a constructive approach. The execution of the schedule is simulated under stochastic resource failures. The comparison results between schedules that are temporally protected and otherwise were then presented.

6.1 Designing the Test Data

6.1.1 Problem Description

We need a set of job shop scheduling problems to test theperformance of Temporal Protection mechanism (namely TP1) within ODO:TNG. Namely, given a scheduling problem and a description of resource failure situations, will our temporally protected schedule survive the uncertainty reality better? As mentioned in section 5.7, ODO:TNG calculates temporal protection using TP1 method which was originally introduced in chapter four.

In [Sadeh 91], a set of 60 scheduling problems were randomly generated to test the performance of Micro-Boss scheduling mechanism. Each problem has 5 resources and 10 jobs of 5 activities each (i.e., a total of 50 activities per problem). Parameters include tight, narrow or wide due date and one or two bottlenecks. So there are six possible parameter settings (denoted as e0ddr1, e0ddr2, enddr1, enddr2, ewddr1 and ewddr2) and ten problems were randomly generated for each parameter configuration.These problems are considered as generic job shop scheduling problems and are often chosen as benchmarking tool for other researches. However, we have not seen any benchmarking data on these problems for scheduling under uncertainty. We decided to choose some of Sadeh's problems as the basis for our experiment design.

We chose only one problem from each of the six groups as the test base. There are going to be dozens of experiments on each base problem and the total number of experiments would be astronomical if all 60 problems were taken. On the other hand, each problem presents the scheduling difficulty pertaining to that group and is fully representative. See appendix A.8 for complete problem description files in Podl. In summary, the six base job shop scheduling problems are respectively referred to as:

- e0ddr1: tight due date, 1 bottleneck.
- e0ddr2: tight due date,2 bottlenecks.
- enddr1: narrow due date, 1 bottleneck.

- enddr2: narrow due date,2 bottlenecks.
- ewddr1: wide due date, 1 bottleneck.
- ewddr2: wide due date,2 bottlenecks.

The scheduling environment is a job shop with 5 machines (1 or 2 of them represent bottleneck(s)). 10 jobs need to be scheduled to meet their assigned due dates. Each job contains 5 sequential activities (i.e., 50 activities in total). Each activity requires one of the 5 machines. No setup is needed on a machine between activities. No preemption is allowed. All resources are subject to failures. A predictive schedule is generated and resources other than required machine will be released according to the schedule. The release time represents the start of the activity's work in process. A shop floor dispatcher decides on the real start-end times of an activity. An activity can not start earlier than its release time even if the machine is available.

We will evaluate the schedule robustness with two criteria: total cost of WIP and tardiness, and schedule deviation (in terms of makespan), defined as:

$$TotalCost = \sum_{i=1}^{50} (Tardiness + WorkInProcess)$$

$$MakespanDeviation = |ScheduledMakespan - ExecutedMakespan|$$

Tardiness and work in process are defined the same way as they were defined in the single machine problems in chapter 4. We assume the unit cost for both tardiness and WIP to be one. We will compare the impact of resource failure on execution cost between temporally protected schedules and unprotected schedules. The main goal is to prove the validity of the temporal protection technique and to relate its performance with problem scenarios.

We have chosen a relatively simple job shop scheduling problem. ODO:TNG can represent and solve much more complex real world job shop scheduling problems. Resource types are differentiated into non-consumable, consumable material and container type. An activity can require conjunctive or disjunctive resources. Disjunctive process plan and dynamic setup times can also be modeled. Our work on resource uncertainty representation adds another aspect to the scheduler's ability to solve real world problems.

6.1.2 Resource Uncertainty Parameters

The choices of random distributions for describing mean time between failure (MTBF) and failure duration (FD) are discussed below. Imposing different distributions on different resources gives rise to various uncertainty scenarios.

MTBF Distributions

The distribution of mean time between failure represents the frequency of failures on the machine. In Sadeh's problem generation, activity durations on the bottleneck machine follow a uniform distribution of U(8,16). The distribution of MTBF can be designed relevant to the average activity duration on the bottleneck machine, which is 12 in all base problems.

We design that $MTBF_{mean}$ has two options in our experiments:

- *Frequent Failure*: $MTBF_{mean}$ equals to the average duration on the bottleneck resource. $MTBF_{mean} = 12$
- *Non Frequent Failure*: $MTBF_{mean}$ equals about 3 times the average duration on the bottleneck resource. $MTBF_{mean} = 35$

Minimum and maximum MTBF depends on the choice of variance levels. We chose two variance values: 20% or 40%. If a 20% deviation is chosen, the corresponding uniform distribution for $MTBF_{mean} = 12$ will be $U(10, 15)$ (Some approximation was done to make it integer). If a 40% deviation is chosen, then the corresponding uniform distribution for $MTBF_{mean} = 12$ will be $U(8, 17)$. Similarly, we have two other distributions when $MTBF_{mean} = 35$: $U(30, 40)$ when variance is 20% and $U(25, 45)$ when variance is 40%. These four distributions are going to be used to describe MTBF:

$U(10, 15)$ $U(8, 17)$ $U(30, 40)$ $U(25, 45)$

FD Distributions

We design the mean of failure duration to be in 25% proportion with the mean of MTBF. It makes sense for the failure time to be overall much less than the production time on a resource. Therefore, we take $FD_{mean} = 3$ when $MTBF_{mean} = 12$, and $FD_{mean} = 9$ when $MTBF_{mean} = 35$. The variance of FD takes the same 2 levels (20% and 40%) as that for MTBF. The four FD distributions that correspond to that of MTBF will be:

$U(2,4)$ $U(1,5)$ $U(7, 11)$ $U(5, 13)$

Describing Resource Failures with MTBF/FD Distributions

MTBF and FD distributions are used together to describe the frequency and outage of failures. Four distribution combinations are to be used in our experiments. They are called D1 to D4 as in Table 11. For instance, D1 describes a resource failure situation where mean time between failures follows a uniform distribution of U(10,15) and failure duration follows another uniform distribution of U(2,4). D2 to D4 can be similarly interpreted.

Table 11. Machine Failure Distributions

	D1	**D2**	**D3**	**D4**
MTBF	$U(10, 15)$	$U(30, 40)$	$U(8, 17)$	$U(25, 45)$
FD	$U(2, 4)$	$U(7, 11)$	$U(1, 5)$	$U(5, 13)$

a. MTBF mean = 12 or 35

b. FD mean = 3 or 9

c. deviation = 20% or 40%

6.1.3 Uncertainty Scenarios and Experiments Design

Each of the test problems has five resources, of which resource 2 is the bottleneck resource in the 1-bottleneck problems (e0ddr1, enddr1 and ewddr1) and resource 2 and 4 are the bottleneck resources in the 2-bottleneck problems (e0ddr2, enddr2 and ewddr2). Each resource could be potentially failure prone and the failure could follow any of D1 to D4 representations.

We are to design two different sets of testing scenarios for 1-bottleneck and 2-bottleneck groups. The intention is to expose the scheduling to all possible resource failure situations, which may include heavy failures or light failures, failures on bottleneck resource(s) only or failures on both bottleneck and non-bottleneck resources. D2 and D4 as defined in Table 11 describe more severe failures than D1 and D3. The choices of distributions and resource(s) that carry the distributions comprise a specific uncertainty scenario. We designed scenarios for one and two bottleneck problems respectively. For each scenario, multiple experiments can be implemented to test the performance of temporal protection technique. The scenarios are listed in Table 12 and 14. Experiments designed are listed in Table 13 and 15 accordingly. Some explanation of the four tables comes in the next few paragraphs.

For the one bottleneck problems, 10 uncertainty scenarios are tested which are listed in Table 12. First four scenarios have Resource 2 (which is the bottleneck resource in the problem) being subjected to failures D1 to D4 respectively. Look at Table 11 and we will know that D2 and D4 are more serious failures than D1 and D3, while D3 and D4 have a bigger variance than their counterparts. Scenarios five to eight select the four non-bottleneck resources to be the uncertain resource which fails according to D1 to D4 respectively. The last two scenarios choose both the bottleneck resource (resource 2) and another non-bottleneck resource (resource 0 and 4 respectively) as critical resources.

Two experiments are designed for each scenario. The first experiment generates the schedule with temporal protection. The second experiment then generates another schedule without temporal protection. Both schedules generated are simulated under the presumed failure distributions. The unprotected schedule is used as a comparison baseline. We are to compare the execution costs of the schedules that are generated with and without temporal protection but are executed under random resource breakdowns. Table 13 lists the experiments design for one bottleneck problems. 20 experiments are designed for the 10 uncertainty scenarios in Table 12. Experiment 1' is used to index the unprotected case for experiment 1 under uncertainty scenario 1. Same indexing holds for other 9 pairs of experiments (2, 2', ... , 10, 10'). (P) after failure description (D1 to D4) means that temporal protection will take this uncertainty into consideration and (NP) means that temporal protection will not protect the specific failure prone resource.

Table 12. Uncertainty Scenarios for 1-bottleneck problems

Uncertainty Scenario	Resource0	Resource1	Resource2	Resource3	Resource4
1			D1[a]		
2			D2		
3			D3		
4			D4		
5	D1				
6		D2			
7				D3	
8					D4
9	D2		D1		
10			D4		D3

a. See Table 12 for definitions of D1-D4.

Table 13. Experiments Design for 1-bottleneck problems

Experiment	Resource0	Resource1	Resource2	Resource3	Resource4
1			D1 (P)[a]		
1'			D1 (NP)[b]		
2			D2 (P)		
2'			D2 (NP)		
3			D3 (P)		
3'			D3 (NP)		
4			D4 (P)		
4'			D4 (NP)		
5	D1 (P)				
5'	D1 (NP)				
6		D2 (P)			
6'		D2 (NP)			
7				D3 (P)	
7'				D3 (NP)	
8					D4 (P)
8'					D4 (NP)
9	D2 (P)		D1 (P)		
9'	D2 (NP)		D1 (NP)		
10			D4 (P)		D3 (P)
10'			D4 (NP)		D3 (NP)

a. Scheduling with temporal protection.
b. Scheduling without temporal protection.

For the 2-bottleneck (Resource 2 and Resource 4) problems, 8 uncertainty scenarios are listed in Table 14. Scenarios one to six have both Resource 2 and 4 subject to failures, while some situations (e.g., scenario 4) present more serious failures than others (e.g., scenario 3). Scenario seven and eight add another non bottleneck resource (Resource 0) as uncertain resource as well. Table 15 lists the corresponding experiments design for 2-bottleneck problems. Note that more than two experiments can be performed for each scenario. For instance, we can test the cases when: both failure resources are protected; one is protected and the other one is not; or both are unprotected. More combinations exist when 3 resources are uncertain (scenario 7 and 8). We designed four experiments for scenarios 1 and 8. Two experiments are designed for all other scenarios.

Table 14. Uncertainty Scenarios for 2-bottleneck problems

Uncertainty Scenario	**Resoruce0**	**Resource1**	**Resource2**	**Resource3**	**Resource4**
1			D1		D2
2			D1		D4
3			D1		D3
4			D2		D4
5			D2		D3
6			D4		D3
7	D4		D1		D1
8	D2		D2		D2

Table 15. Experiments Design for 2-bottleneck problems

Experiment	Resource0	Resource1	Resource2	Resource3	Resource4
1			D1 (P)[a]		D2 (P)
1'			D1 (NP)[b]		D2 (P)
2			D1 (P)		D2 (P)
2'			D1 (NP)		D2 (NP)
3			D1 (P)		D4 (P)
3'			D1 (NP)		D4 (NP)
4			D1 (P)		D3 (P)
4'			D1 (NP)		D3 (NP)
5			D2 (P)		D4 (P)
5'			D2 (NP)		D4 (NP)
6			D2 (P)		D3 (P)
6'			D2 (NP)		D3 (NP)
7			D4 (P)		D3 (P)
7'			D4 (NP)		D3 (NP)
8	D4 (P)		D1 (P)		D1 (P)
8'	D4 (NP)		D1 (NP)		D1 (NP)
9	D2 (P)		D2 (P)		D2 (P)
9'	D2 (P)		D2 (NP)		D2 (P)
10	D2 (P)		D2 (P)		D2 (P)
10'	D2 (NP)		D2 (NP)		D2 (NP)

a. Scheduling with temporal protection.

b. Scheduling without temporal protection.

6.2 Experiment Steps

Each experiment is performed in 2 steps: First a predictive schedule is generated by ODO:TNG. The schedule execution is then simulated under random resource failures.

6.2.1 Generate Schedule

Chapter five introduced the problem solving structure of ODO:TNG. Figure 33 highlights the main components of the solving process.

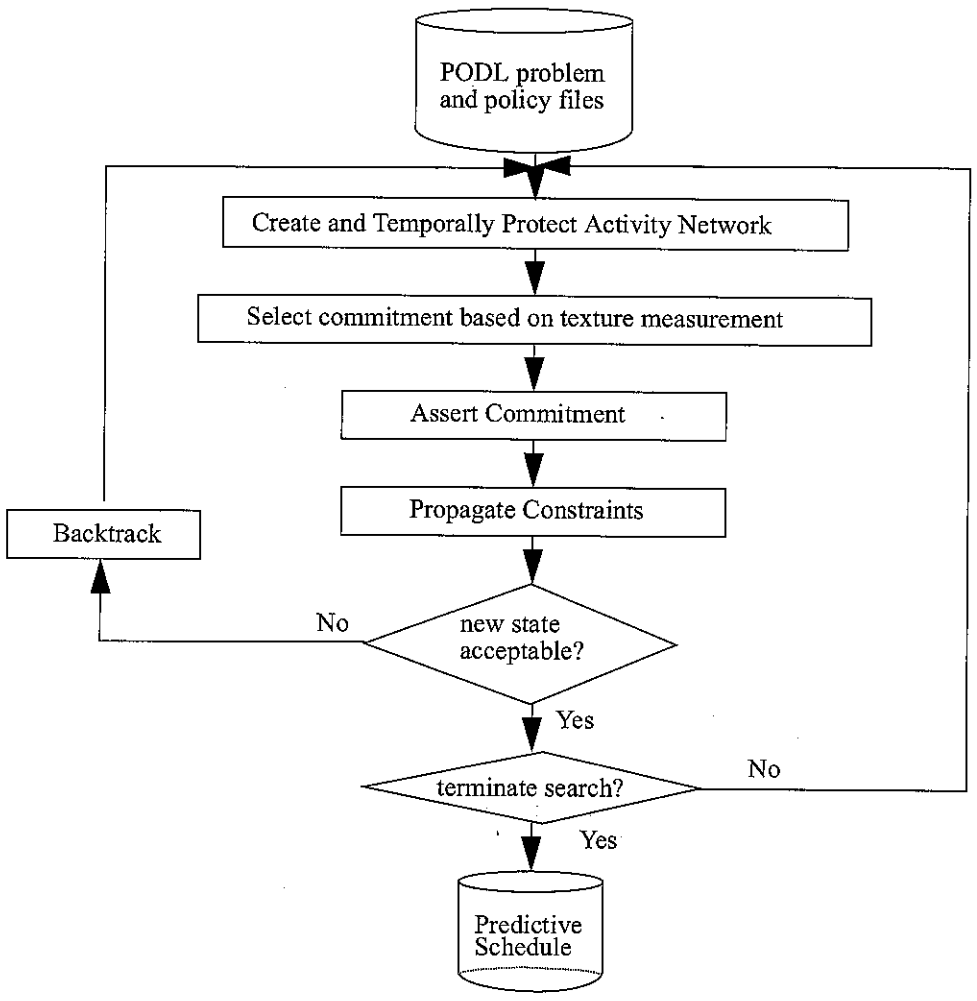

PODL problem and policy files
Create and Temporally Protect Activity Network
Select commitment based on texture measurement
Assert Commitment
Propagate Constraints
Backtrack
No
new state acceptable?
Yes
terminate search?
No
Yes
Predictive Schedule

Figure 33. Problem Solving Process

6.2.1.1 Choice of Policy

A policy in ODO:TNG is defined as a specification of the exact manner in which each component of the above search process is to be performed. It is a ODO specific naming and is generally referred to as texture or heuristics elsewhere. ODO:TNG structures the solving process into AtomicPolicy and MetaPolicy. A MetaPolicy may contain a serial of AtomicPolicies. A new policy can begin once a previous policy has ended. Each AtomicPolicy defines a set of filter-functions that specify the manner in which variable/value are selected and the assertion is propagated. It also defines the backtrack mechanism, termination criteria, and other structures. The set of filter functions are fed by the users.

We use the *resource contention* texture which was researched and experimented by [Sadeh 91]. Based on a probabilistic demand profile estimation, the most contended resource is dynamically identified and the activity which contributes the most to that demand is chosen. From the domain of the activity's start time variable, the value which has the highest probability of surviving resource contention is asserted. The corresponding variable and value selection heuristics are called ORR and FSS by Sadeh respectively and details of the resource contention texture can be found in [Sadeh 91].

This choice of texture suits the problem solving need of temporally protected activity network. Since the demand profile is constructed based on the activities' extended duration and thus reflects the expected extra demand for the resource due to failure interruptions.

ORR and FSS are implemented as policy options in ODO:TNG. The PODL description of the start time assignment policy which uses demand based texture follows. Resource assignment is simple in our case since the test problems do not have alternative resource choices for all the activities. The only resource being requested is assigned to the activity. On the other hand, we do assume a activity independent lump-sum resource (could be material, fixture, etc.) to be released and consumed by each activity. They are considered reliable resources and do not fail. But once they are scheduled to be released, they contribute to the WIP cost until the activity is completed. Note that resource assignment policy precedes start time assignment policy in the current ODO:TNG policy structure.

```
(atomic-policy :name construct
      :commitment-type assign-starttime
      :forward-commitments
         (filters :generate (unassigned orr)
      :select (fss)
      :score none)
      :backward-commitments
         (filters :generate (most-recent-failure))
      :propagation-methods (temporal-possible-value unit-resource)
      :backtrack-method chronological
      :termination-criteria (cost == 0 || search-iterations == 1000)
      :state-cost-function num-unassigned-activities
      :state-acceptance-criteria no-empty-pvs)
```

6.2.1.2 Output of Predictive Schedule

The searching stops when all activities are scheduled (start time assigned) or the search iterations reach 1000 **:termination-criteria (cost == 0 || search-iterations == 1000)**. The output is a text schedule which describes the activities' resource release time, start/end times, duration and resource request. An excerpt of a schedule generated may look like the following. Note that non-critical resource is released earlier than the activity's planned start time (not necessarily true for all activities).

```
A02: 116..[117..117-16..16-->133..133]
 Resource Requests:
   [RRV: NamedObject_61
      Resource Role: machine
      Amount Required: 1
      Possible Domain:
         Resource: Resource2
   ]

A12: 68..[69..69-17..17-->86..86]
 Resource Requests:
   [RRV: NamedObject_116
      Resource Role: machine
      Amount Required: 1
      Possible Domain:
         Resource: Resource2
```

```
    ]

A32: 14..[16..16-18..18-->34..34]
 Resource Requests:
   [RRV: NamedObject_226
      Resource Role: machine
      Amount Required: 1
      Possible Domain:
         Resource: Resource2
   ]
```

6.2.2 Simulate Schedule Execution

The simulation mechanism was described at the end of chapter five. It is an important addition to ODO:TNG and a crucial part of our experiments.

Given a schedule generated and an uncertainty scenario, we can simulate its execution by randomly generate failures on the uncertain resource according to its MTBF and FD distributions. We can then examine the performance of the predictive schedule confronted by the failure uncertainty. Our experiments carry out simulation for schedules generated both with and without temporal protection.

The PODL command of

```
(simulate)
```

starts the simulation of the generated schedule. A list of failure intervals are randomly generated according to the current uncertainty scenario description in the resource section of Podl problem description file. These failures will interrupt or delay the scheduled activities. A constraint based simulator (see end of chapter five) decides the real start and end times for each activity based on the previous finish time on the resource and the planned release time for the current activity. The dispatcher may decide to start the activity earlier than its planned start time if the previous activity finishes sooner. The assertion of real start time is propagated to reachieve the network consistency. When all activities are executed, a brief report of the simulation results such as the below is generated. The total cost of schedule execution is the sum of work in process cost and tardiness cost for all activities, assuming the unit costs per time unit of WIP and tardiness are both 1.

```
******************************************
********** Simulation Report *************
******************************************
Problem #: e0ddr1-fail1

Failures on Resource2---- MTBF [10 15] FD [2 4]
[10 12][22 24][34 36][46 48][58 60][70 72][82 84][94 96][106 108][118
120][130 132][142 144][154 156][166 168][178 180][190 192]

Scheduled Makespan ---- 185
Running Makespan ---- 185

Total WIP cost ---- 451
Total Tardiness cost ---- 74
# of Late Activities ---- 4
Total Cost ---- 525
```

```
Activities start as planned ---- 1
Activities start early ---- 45
Activities start late ---- 4
########## END OF SIMULATION FOR 'e0ddr1-fail1' ##########
```

The simulation of unprotected schedule under the same scenario generates the following report. Note that the total cost of executing the schedule increased significantly due to bigger WIP cost.

```
*****************************************
********** Simulation Report *************
*****************************************
Problem #: e0ddr1-fail1u

Failures on Resource2---- MTBF [10 15]FD [2 4]
[15 19][34 38][53 57][72 76][91 95][110 114][129 133][148 152][167 171][186
190]

Scheduled Makespan ---- 150
Running Makespan ---- 185

Total WIP cost ---- 1018
Total Tardiness cost ---- 88
# of Late Activities ---- 4
Total Cost ---- 1106

Activities start as planned ---- 0
Activities start early ---- 20
Activities start late ---- 30
########## END OF SIMULATION FOR 'e0ddr1-fail1u' #########
```

6.2.3 Number of Observations

For each of the six base problems, 10 protected schedules and 10 unprotected schedules are to be simulated according to the experiment design in Table 13 and Table 15. Each of the experiments is run ten times to reflect the random performance of resource failures. There are altogether 6x20x10=1200 simulation results to be collected and analyzed accordingly.

6.3 Simulation Results

Table 16 to 21 summarize the simulation results for all simulation runs.

Table 16. Simulation results for problem e0ddr1 (tight due date, 1 bottleneck)[a] - ORR

Experiment	Total Cost	WIP Cost	Tardiness	Late Activities	Scheduled Makespan	Running Makespan
1	589.1	502.9	86.2	4.2	185	188.7
1'[b]	913.5	855.5	58	3.2	150	175.2
2	650.2	551	99.2	4.6	185	192.4
2'	1008.8	934.4	74.4	3.7	150	180.4
3	558.8	468.3	90.5	4.2	185	188
3'	985.9	915.6	70.3	3.4	150	178.2
4	845.2	709.8	135.4	5	185	202.5
4'	948.7	885.6	63.1	3.1	150	177.4
5	445	443	2	1	151	151
5'	451	448.4	2.6	1	150	151.6
6	483.8	468.8	15	2	157	157
6'	454.4	453.4	1	1	150	150
7	449.8	448.8	1	1	150	150
7'	465.4	464.4	1	1	150	150
8	667	625.4	41.6	3.6	157	165.4
8'	601.8	585.8	16	2	150	157.2
9	627.7	537.2	90.5	4	189	193.3
9'	976.3	910.2	66.1	3.6	150	179.9
10	770.1	668	102.1	4.4	179	187.5
10'	1192.4	1093.5	98.9	4.1	150	186.1

a. Each value in the cell is the average of 10 simulation run results.

b. 1' is the unprotected case of 1.

Table 17. Simulation results for problem e0ddr2 (tight due date,2 bottleneck) - ORR

Experiment	Total Cost	WIP cost	Tardiness	Late Activities	Scheduled Makespan	Running Makespan
1	901.5	577.3	324.2	5.1	263	268.1
1'	1162.1	868.4	293.7	5	241	259.1
2	980.5	644.8	335.7	5.5	263	268.7
2'	1197.2	889	308.2	5.2	229	264.1
3	954.6	620.8	333.8	5.6	263	270.5
3'	1216.1	903.7	312.4	5.3	229	264.2
4	911.9	595.5	316.4	5	263	265.1
4'	1191.4	886.7	304.7	5.2	229	262.7
5	1101.5	736.4	365.1	5.7	263	276.3
5'	1318.6	985.2	333.4	5.4	229	270.2
6	1100.8	746.4	354.4	5.6	263	272.9
6'	1254.3	942.3	312	5.1	229	262.4
7	1076	726.8	349.2	5.6	263	271.3
7'	1300.3	975.5	324.8	5.4	229	267.2
8	1008.7	619.7	389	6	276	278.2
8'	1416.3	1082.5	333.8	5.5	229	268.7
9	1137.6	717.8	419.8	6	276	283.7
9'	1495.1	1128.2	366.9	6	248	274.1
10	1143.5	727.2	416.3	6	276	281.9
10'	1421	1071.6	349.4	5.3	229	271.2

Table 18. Simulation results for problem enddr1 (narrow due date,1 bottleneck) - ORR

Experiment	Total Cost	WIP cost	Tardiness	Late Activities	Scheduled Makespan	Running Makespan
1	586.3	483.5	102.8	3.2	189	191.3
1'	751.6	706.7	44.9	3	152	172.5
2	715.4	593.4	122	3.6	189	196.7
2'	834.1	776.6	57.5	3.2	152	175.9
3	602.4	495.8	106.6	3.2	189	192.3
3'	774.8	726	48.8	3.1	152	173.9
4	738.5	609.7	128.8	3.5	189	198.4
4'	936.5	862.8	73.7	3.2	152	183.3
5	449.1	444.1	5	1	152	152
5'	484.8	478.8	6	1	152	153
6	612.6	598.1	14.5	1.5	156	161.4
6'	707.8	691.9	15.9	2	152	160.8
7	502.2	477.6	24.6	1.1	172	172.9
7'	589.6	577.9	11.7	1.4	152	156.7
8	508.4	503.4	5	1	152	152
8'	484	479	5	1	152	152
9	644.4	538.9	105.5	3.6	189	191.5
9'	806.9	753	53.9	3	152	174.8
10	750.8	647	103.8	3.6	195	204.4
10'	914.6	859.4	55.2	3.2	152	178.1

Table 19. Simulation results for problem enddr2 (narrow due date, 2 bottleneck) - ORR

Experiment	Total Cost	WIP cost	Tardiness	Late Activities	Scheduled Makespan	Running Makespan
1	886.2	611.7	274.5	5.2	262	269.4
1'	1221.8	921.8	300	5	251	271.3
2	895.4	623.1	272.3	5.3	262	268.9
2'	1267.6	1009.2	258.4	5	225	263.9
3	895.2	626.6	268.6	5.1	262	267.2
3'	1259.7	1003.7	256	5	225	264
4	879.4	619.4	260	5	262	265.1
4'	1263.2	1010.7	252.5	5	225	262.2
5	935.3	658.4	276.9	5.3	262	267.3
5'	1553.1	1233.7	319.4	5.2	225	278.2
6	999.5	710.9	288.6	5.4	262	271.1
6'	1530.3	1218.9	311.4	5.3	225	275.6
7	989.5	706.9	282.6	5.3	262	269.2
7'	1524.6	1217.7	306.9	5.2	225	273.7
8	1002.7	622.8	379.9	6	281	283.7
8'	1526.2	1246.7	279.5	5	225	268.2
9	1133.6	735	398.6	6	281	287.8
9'	1427.4	1107.2	320.2	6	249	273.9
10	1260.9	830.4	430.5	6	281	294.4
10'	1526.3	1232.1	294.2	5.4	225	272.6

Table 20. Simulation results for problem ewddr1 (wide due date, 1 bottleneck) - ORR

Experiment	Total Cost	WIP cost	Tardiness	Late Activities	Scheduled Makespan	Running Makespan
1	537.3	481.2	56.1	3.6	187	189.1
1'	822.3	736.4	85.9	4	166	189.2
2	624.9	550.2	74.7	4.6	187	194.4
2'	854.1	760.9	93.2	4	166	190.6
3	609.7	537.9	71.8	4.2	187	193.3
3'	813.7	730.2	83.5	4.2	166	188.6
4	527.3	471.8	55.5	4.2	187	189.3
4'	1030	905.2	124.8	4.2	166	199.7
5	467.9	443.9	24	1	167	167
5'	461	444.4	16.6	2.2	166	167.4
6	611.8	601.8	10	2	162	167.5
6'	762.4	716	46.4	4	166	175
7	473.5	473.5	0	0	149	150.1
7'	642.6	606.9	35.7	3.7	166	170.8
8	487.6	471	16.6	2	166	167.6
8'	559.9	542.1	17.8	2	166	168.8
9	656.4	571.3	85.1	4	188	192.5
9'	928.2	831.1	97.1	4	166	194
10	712	627.4	84.6	4.1	187	196.4
10'	818.7	751	67.7	4	166	184.2

Table 21. Simulation results for problem ewddr2 (wide due date, 2 bottleneck) - ORR

Experiment	Total Cost	WIP cost	Tardiness	Late Activities	Scheduled Makespan	Running Makespan
1	851.2	618.6	232.6	4.6	257	263.7
1'	1244	958.3	285.7	6	251	276.7
2	832.4	604.8	227.6	4.7	257	262.4
2'	1195.3	936.4	258.9	5.7	233	273.1
3	854.9	623.9	231	4.8	257	263.6
3'	1266.8	986.1	280.7	5.9	233	279.5
4	816	595.5	220.5	4.9	257	261
4'	1172.4	923.6	248.8	5.6	233	269.2
5	912	672.4	239.6	4.9	257	264.6
5'	1367.1	1074.7	292.4	6	233	279.3
6	987.7	728.4	259.3	5	257	269.5
6'	1380.7	1083.5	297.2	5.9	233	279
7	1077.3	803	274.3	5.1	257	270.5
7'	1273.8	999.1	274.7	5.9	233	275
8	1056.8	790.2	266.6	5.7	265	271.4
8'	1396.7	1130.3	266.4	6	233	273.7
9	1644.8	1297.4	347.4	6	251	290.3
9'	1273.4	1085.3	188.1	5	226	246.1
10	1550	1224.7	325.3	5.8	251	289.3
10'	1685.9	1351.5	334.4	5.8	233	287.8

We can visualize the difference in total cost between protected schedules and unprotected schedules from figures 34 to 39.

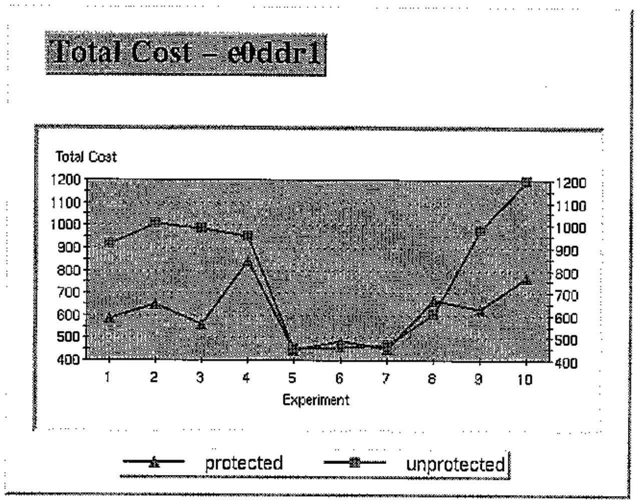


Figure 34. Simulation Result - Total Cost for e0ddr1.

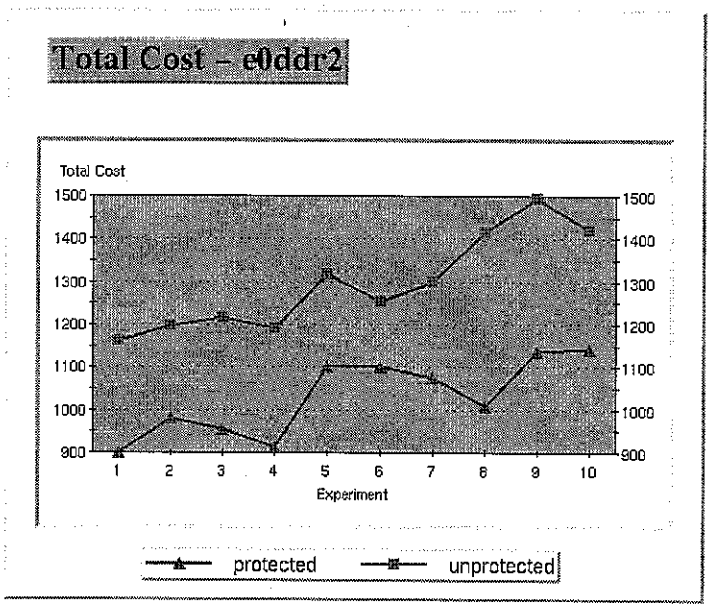


Figure 35. Simulation Result - Total Cost for e0ddr2.

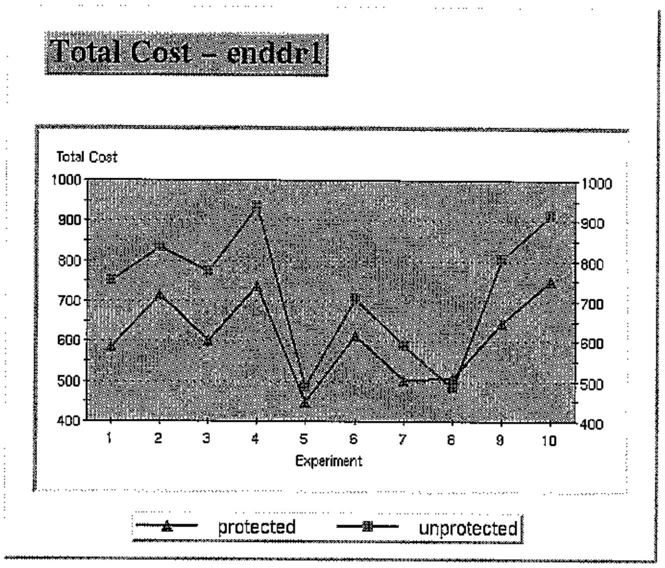


Figure 36. Simulation Result - Total Cost for enddrl.

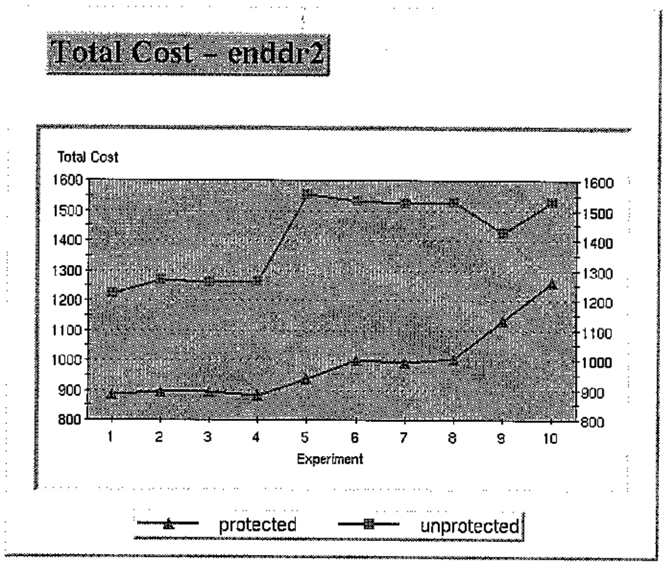


Figure 37. Simulation Result - Total Cost for enddr2.

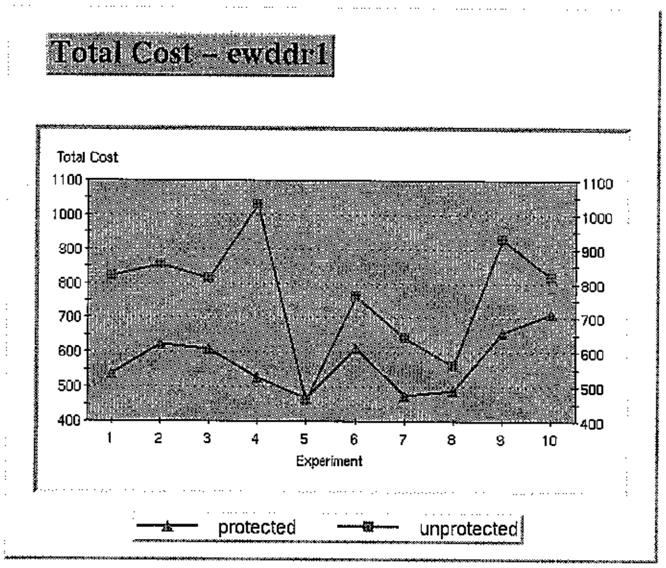


Figure 38. Simulation Result - Total Cost for ewddr1.

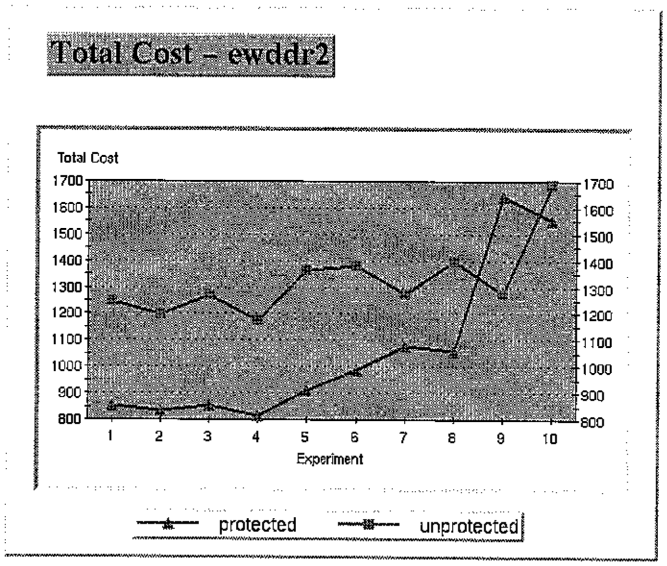


Figure 39. Simulation Result - Total Cost for ewddr2.

6.4 Result Analysis

Observing the total cost across all the experiments, it is obvious that each schedule protected by temporal protection performs better or equally well than their unprotected counterpart in simulation (the cost curve of unprotected schedules is above protected curve). Table 22 summarizes the cost savings by problem category. The overall cost saving is around 23%. The saving for 2-bottleneck problems is slightly higher than that for 1-bottleneck problems.

Table 22. Average Cost Comparison

Problem	**Average Cost (protected)**	**Average Cost (unprotected)**	**% of cost saving**
e0ddr1	609	800	23.9%
e0ddr2	1032	1297	20.5%
enddr1	611	728	16.1%
enddr2	988	1410	30.0%
ewddr1	571	769	25.8%
ewddr2	1058	1326	20.2%
tight due date	820	1049	21.8%
narrow due date	800	1069	25.2%
wide due date	814	1048	22.3%
1-bottleneck	597	766	22.1%
2-bottleneck	1026	1344	23.7%
grand mean	*811*	*1055*	*23.1%*

Table 23 compares the schedule deviation criterion. For each problem, it shows its average deviation between scheduled makespan and execution makespan. The average execution makespan is given for calculating the percentage of deviation. Data for both protected experiment and unprotected one is compared. It is obvious that deviation averages only 3% when temporal protection is applied compared to 12% when the schedule is not protected. A small schedule deviation propagate less uncertainty to adjacent scheduling jobs and is certainly desirable.

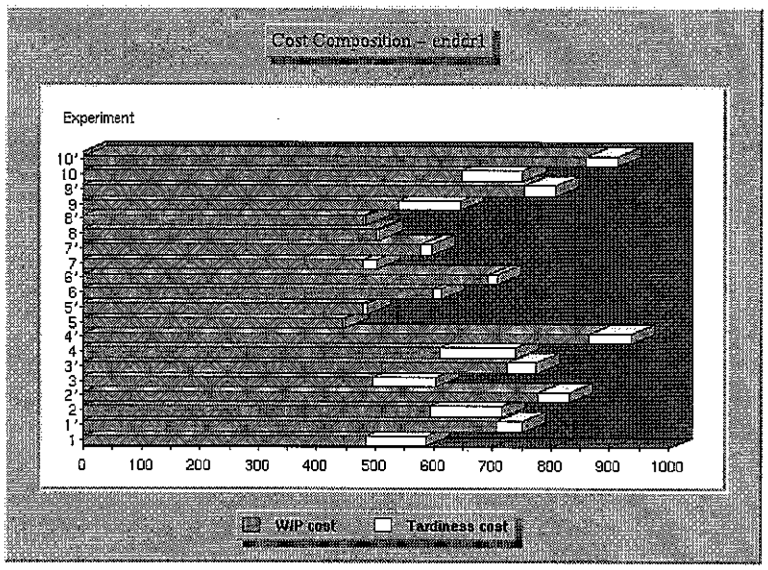
Table 23. Results - Makespan Deviation Comparison

Problem	Protected deviation	Protected makespan	Protected %	Unprotected deviation	Unprotected makespan	Unprotected %
e0ddr1	5.3	181.1	2.9	16.1	168.2	10.0
e0ddr2	6.8	274.3	2.5	34.3	270.4	12.7
enddr1	4.1	181.3	2.3	16.1	168.2	10.0
enddr2	6.7	274.2	2.4	40.4	270.4	15.0
ewddr1	4.0	180.6	2.2	16.8	182.9	9.2
ewddr2	14.0	270.5	5.2	38.0	274.1	13.9
tight DD	6.1	227.7	2.7	25.2	219.3	11.5
narrow	5.4	227.8	2.4	28.3	219.3	12.9
wide	9.0	225.5	4.0	27.4	228.5	12.0
1-btnk	4.5	181	2.5	16.3	173.1	9.4
2-btnk	9.2	273	3.4	37.6	271.6	13.8
mean	*6.9*	*227*	*3.0*	*27.0*	*222.4*	*12.1*

Some observations can be drawn from the simulation results:

- Temporal protection reduces the WIP cost and increases the tardiness cost of executing a schedule. Due to the duration extension of temporal protection, tardiness cost for protected schedule are inevitably higher than that of unprotected schedule (Figure 40). Exceptions exist for problems ewddr1 and ewdrr2 (Figure 41), when due dates are wider and tardiness cost tends to be insignificant. Under this case, the tardiness cost of protected schedules is generally less than their unprotected counterpart. We can examine the impact of duration extension on makespan as well. For protected schedules, the actual execution makespan runs very close to or sometimes the same as the scheduled makespan. In the unprotected case, the execution makespan is much longer than the scheduled one since the scheduling did not take the machine failure into account. The ability to complete schedule around expected time is very much desired. Because the deviation in one schedule's completion time can be carried over to the following schedules. The smaller deviation there exists locally, the better chance that following schedule can start as planned.

Figure 40. Cost Composition - enddr1



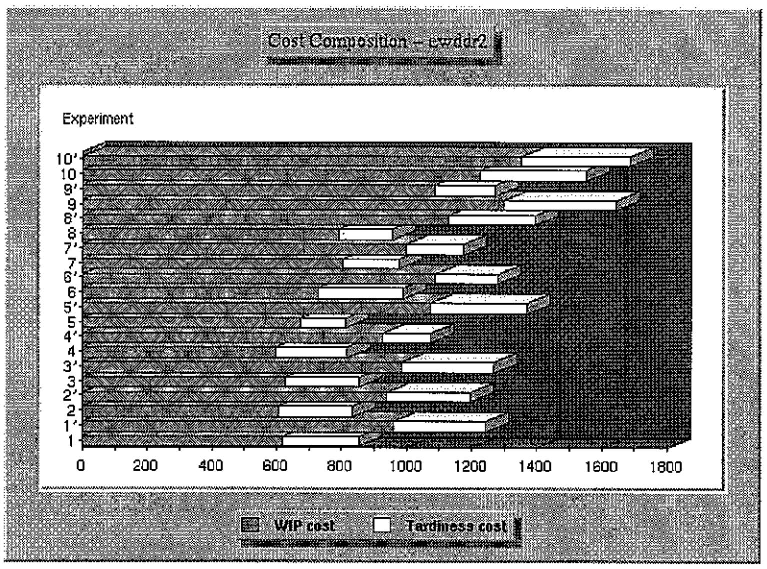


Figure 41. Cost Composition - ewddr2

- WIP cost contributes the most part of the total cost (80+%). Temporal protection reduces WIP cost by releasing the activity around the time that it can be worked on. Therefore, although tardiness cost grows when schedule is protected, the saving in WIP cost has a greater effect on the final cost reduction.

- Cost grows when machine failure uncertainty grows. Longer and more frequent failures increase both WIP cost (activity lasts longer) and potential tardiness (activity finishes late). This explains why total cost for 2-bottleneck problems are generally much higher than that for 1-bottleneck problems. Extra amount of failures prolong the schedule and increase the costs. Same explanation applies to the higher cost caused by scenario D2 or D4 than that by D1 or D3 (see experiment 1 to 4 in e0ddr1 for example). Failure situation is more uncertainty when, failures happen on bigger number of machines, average mtbf is small and/or average failure duration is big, or, the deviation from mean value is bigger for MTBF and FD distributions. i.e., $U(25, 45)$ is more uncertain than $U(30, 40)$, since it has a bigger range of deviation.

- The cost saving by temporal protection is more significant when the bottleneck machine is subject to failure. In the case of one bottleneck problems (Figure 34, 36 and 38), Scenario 1 to 4 fail on the bottleneck machine R2. Scenario 5 to 8 in the same figures have failures happen on R0, R1, R3 and R4 respectively. The saving in execution costs by temporal protection for scenario 5 to 8 is trivial, if not none (average total cost saving due to protection is only about 1% for scenario 5 to 8, in problem e0ddr1). On the other hand, scenario 1 to 4 result in obvious differences whether temporal protection is supplied or not. In problem e0ddr1, they have an average cost saving of 33.2%.

- For one bottleneck problems, scenarios 9 and 10 are extensions of scenarios 1 and 4. An additional failure was imposed on a non bottleneck machine in addition to the original failure on bottleneck. The results show that no big impact does the additional failure have on the total cost. Double failures generate slightly higher cost than single failure case in both e0ddr1 and enddr1, but lower in ewddr1. No pattern exists.

- Temporal protection works well under bigger variance of distributions. Take scenario 6 and 7 in 2-bottleneck problems, for example, they are similar except that scenario 7 uses a 40% deviation ($U(25, 45)$) for R2 failure distribution, while scenario 6 uses a 20% deviation ($U(30, 40)$). The results from e0ddr2 show that scenario 7 saves more in total cost than scenario 6. i.e., while the environment is getting more uncertain, the protection mechanism is addressing the dynamics and actually generates schedule that performs robustly under the environment. This attributes to the earlier release time by temporal protection, which is even earlier if the uncertainty is bigger. The flexibility of starting as early as the release time reduces the waiting time of activity, and thus its WIP cost.

- The tightness of due date does not directly impact the usage of temporal protection too much. Table 23 shows that although tight due date type saves less in terms of total cost, the percentages of cost saving for three types of due date are not significantly different.

Chapter 7 Conclusions and Future Research

7.1 Conclusions

In this thesis, we reviewed the uncertainty factors in the manufacturing environment. The impact of uncertainty on job shop scheduling was discussed. We presented a temporal protection approach as an attempt to deal with resource failure uncertainty in job shop scheduling. The approach focuses on predictively building robust schedules based on knowledge of uncertainty in advance. Activity network is protected so that the schedule generated is less likely to fail while being executed under the failure prone environment. It also allows micro reactions during the schedule execution. The method addresses machine failure uncertainty primarily, but it should be extensible to any probabilistic uncertainty which affects the activity execution directly.

We also implemented the technique within the framework of a generic constraint-based scheduler ODO:TNG. Problem description language PODL was extended to input machine uncertainty information, which are used in calculating temporal protection during the scheduling. A constraint based simulation mechanism was developed to test the robustness of a generated schedule under dynamic failure environment. We used total cost of work in process and tardiness as the criteria for evaluating schedule robustness. Our experimental results showed that temporal protected schedules can be effectively executed under dynamic machine failures, incurring less cost than the unprotected schedules implemented under the same uncertain scenario. The amount of saving depends on the failure pattern and the criticality of the failure machine.

7.2 Future Work

This thesis represents an initial step to integrate uncertainty management into job shop scheduling. Predictive prevention is advocated for dealing with 'known' domain uncertainties. Our first effort was devoted to representing probabilistic failure uncertainty and building robust schedules under machine failure.

Future research can be investigated in many possible channels:

- Protect network neighborhood - Current temporal protection only protects the activity nodes in the constraint graph which relies on unreliable machines. Research can be done to include the neighboring nodes which have direct/indirect temporal relationships with the protected activity. Protection to their release times should provide more flexibility to the schedule execution. More experiments are needed to testify the premise.
- Dealing with other uncertainties in the job shop - As chapter one overviewed, there are other uncertainties on the shop floor, such as random material defects, uncertain rework rate, unreliable staff and human intervention. Certain aspects may be crucial to a specific real world scheduling system. Incorporating the representation and problem solving for these uncertainty situations into the constraint scheduling approach remains to be researched.
- Use reactive scheduling to respond to 'unknown' uncertainties - When uncertainty can not be mathematically modeled, it is very hard to be prevented in advance. Rescheduling is inevitable. Choices are between total rescheduling and heuristic based partial repair.
- Introducing Belief into a Constraint Model - Introducing subjective belief into a constraint (temporal or resource) represents the scheduler's degree of commitment in a particular constraint when he is not 100 percent sure about it. Introducing subjective belief into possible values of a variable symbolizes the scheduler' belief of the goodness of the values. But network propagation has to be updated to be able to propagate with belief values.
- Interaction with the User - Human scheduler should be involved in the decision making process, especially when there exist verbal uncertainties. Under the circumstance when no feasible schedule could be found, it should be possible to relax some constraints whose user belief is under a certain belief threshold. This relaxation is done at the cost of risk taking. The more discreet the scheduler is, the lower threshold is set, the less risk can be taken. The level of risk taking can be set by the user, such that the belief extent to which constraints are no longer relaxable depends on the users' willingness to take the risk of real occurrence of the uncertain events. An example of interaction could be posting the highest failure rate the user can accept for a resource. If this constraint is violated in reality, the resource is so unreliable that it will not be used in the schedule. Its capacity till the end of the scheduling is considered to be zero. The user may want to schedule maintenance activity for the resource.

Bibliography

[Allahverdi 94] Allahverdi, Ali and Mittenthal, John. Scheduling on M Parallel Machines Subject to Random Breakdowns to Minimize Expected Mean Flow Time. *Naval Research Logistics.* **41** 667-682, 1994.

[Allen 84] Allen, J. Towards a General Theory of Action and Time. *Artificial Intelligence.* **23** 123-154, 1984.

[Anthony 65] Anthony, R. N. *Planning and Control Systems: A Framework for Analysis.* Harvard University Graduate School of Business Administration, 1965.

[Anupindi 93] Anupindi, R. and Akella, R. Diversification Under Supply Uncertainty. *Management Science.* **39** (8), August, 1993.

[Aykac 89] Aykac, A. and Corstjens, M. and Gautschi, D. and Horowitz, I. Estimation Uncertainty and Optimal Advertising Decisions. *Management Science.* **35** (1):42-50, January, 1989.

[Baker 74] Baker, K.R. *Introduction to Sequencing and Scheduling.* John Wiley & Sons, 1974.

[Bassok 91] Bassok, Y. and Akella, R. Ordering and Production Decisions with Supply Quality and Demand Uncertainty. *Management Science.* **37** (12):1556-1574, December, 1991.

[Bensana 93] Bensana, Eric and Sicard, Marc. A Constraint Based System for Schedule Construction, Maintenance and Revision. *IJCAI-93 Workshop on Knowledge Based Production Planning, Scheduling, and Control*, pages 11-22. 1993.

[Bonissone 87] Bonissone, P.P. Reasoning, Plausible. *Encyclopedia of Artificial Intelligence.* In Shapiro, S.C., New York, John Wiley, 1987.

[Brennan 93] Brennan, L. and Gupta, S.M. A Structured Analysus of Material Requirements Planning Systems Under Combined Demand and Supply Uncertainty. *International Journal of Production Reseach.* **31** (7):1689-1707, 1993.

[Browne 84] Browne, J. Classification of Flexible Manufacturing Systems. *The FMS Magazine.* **2** (2), 1984.

[Buzacott 82] Buzacott, J.A. The Fundamental Principles of Flexibility in Manufacturing Systems. *Proceedings of the 1st International Conference on Flexible Manufacturing Systemes*, pages 13-22. 1982.

[Chiang 90] Chiang, W.Y. and Fox, M.S. Protection Against Uncertainty in a Deterministic Schdule. *Proceedings of International Conference on Expert Systems for Production and Operations Management.* May, 1990.

[Choong 87] Choong, Y.F. and Gershwin, S.B. A Decomposition Method for the Approximate Evaluation of Capacitated Transfer Lines with Unreliable Machines and Random Processing Times. *IEE Transactions.* **19** (2):150-159, June, 1987.

[Chris 95] Chris, B. and Aggarawal, S. and Gao, H. *PODL Manual.* Technical Report, Enterprise Integration Laboratory, Industrial Engineering Department, University of Toronto, 1995.

[Ciarallo 94] Ciarallo, F.W. and Akella, R. and Morton, T.E. A Periodoc Review, Production Planning Model with Uncertain Capacity and Uncertain Demand - Optimality of Extended Myopic Policies. *Management Science.* **40** (3), March, 1994.

[Cohen 85] Cohen, P.R. *Heuristic Reasoning about Uncertainty: an Artificial Intelligence Approach.* Pitman Advanced Pub., Boston, 1985.

[Cohen 89] Cohen, P.R. Steps Towards Programs that Manage Uncertainty. Kanal, J.F. and Levitt, T.S. and Lemmer, L.N. (editor), *Uncertainty in Artificial Intelligence*Elsevier Science Publishers, 1989.

[Davis 94] Davis, E. *ODO: a constraint-based scheduler founded on a unified problem solving model.* Master's thesis, University of Toronto, 1994.

[Drummond 94] Drummond, M. and Swanson, K. and Bresina, J. Robust Scheduling and Execution for Automatic Telescopes. *Intelligent Scheduling.* In Zweben, M. and Foc, M.S., Morgan Kaufmann, 1994.

[Fargier 94] Fargier, H. and Lang, J. and Martin-Clouaire, R. and Rellier J.P. Uncertainty and Flexibility in Constraint Satisfaction: A Case Study and an Application to Agricultural Planning. Bessiere, C. and Schiex T. (editor), *Proceedings of the ECAI'94 Workshop on Practical Applications Raised by Constraint Satisfaction.* 1994.

[Foote 92] Foote, B.L. and Pulat, P.S. and Badiru, A.B. A Framework for Integrated Production Planning and Scheduling in a Hybrid Assembly Job Shop Environment Under Uncertainty. *2nd Industrial Engineering Research Conference Proceedings,* pages 853-857. 1992.

[Fox 89] Fox, M. and Sadeh, N. and Baykan, C. Constrained Heuristic Search. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence.* 1989.

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

[Fox 84] Fox, M.S. and Smith, S.F. ISIS: A Knowledge-based System for Factory Scheduling. *International Journal of Expert Systems.* **1** (1), 1984.

[Gao 95] Gao, H. and Fox, M.S. and Chiang, W. *Building Robust Schedules: An Empirical Study of Single Machine Scheduling with Uncertainty.* Technical Report, Enterprise Integration Laboratory, Industrial Engineering Department, University of Toronto, 1995.

[Hartl 92] Hartl, R.F. Optimal Acquisition of Pollution Control Equipment Under Uncertainty. *Management Science.* **38** (5):609-622, May, 1992.

[Holthausen 82] Holthausen, D.M. Jr. and Assmus, G. Advertising Budet Allocation Under Uncertainty. *Management Science.* **28** (5):487-499, May, 1982.

[Hong 92] Hong, Yushin and Glassey, C. Roger and Seong, D. The Analysis of a Production Line with Unreliable Machines and Random Processing Times. *IIE Transactions.* **24** 77-83, 1992.

[Hutchinson 94] Hutchinson, G.K. and Pflughoeft, K.A. Flexibile Process Plans: Their Value in Flexible Automation Systems. *International Journal of Production Reseach.* **32** (3):707-719, 1994.

[Kaufmann 84] Kaufmann, A. and Gupta, M.M. *Introduction to Fuzzy Arithmetic: Theory and Applications.* Van Mostrand Reinhold Co., New York, 1984.

[Krause 93] Krause, Paul. *Representing uncertain knowledge : an artificial intelligence approach.* Kluwer Academic Publishers, Dordrecht ; Boston, 1993.

[Law 91] Law, A.M. and Kelton, W.D. *Simulation Modeling and Analysis.* McGraw-Hill Inc., 1991.

[Leon 94] Leon, V.J. and Wu, S.D. and Storer, R.H. Robustness Measures and Robust Scheduling for Job Shop. *IIE Transactions.* **26** (5):32-43, September, 1994.

[Lewis 94] Lewis, E.E. and Chen, Hsin-Chieh. Load-Capacity Interference and the Bathtub Curve. *IEEE Transactions on Reliability.* **43** (3), September, 1994.

[Mamer 87] Mamer, J.W. and Mccardle, K.F. Uncertainty, Competition, and the Adoption of New Technology. *Management Science.* **33** (2):161-177, February, 1987.

[Masood 90] Masood, A. Uncertainty Mangement in Computer Aided Process Planning Systems. *1990 International Industrial Engineering Conference Proceedings*, pages 39-44. 1990.

[Mittenthal 93] Mittenthal, John and Raghavachari, M. Stochastic Single Machine Scheduling with Quadratic Early-Tardy Penalties. *Operations Research.* **41** 786-796, 1993.

[Montgomery 91] Montgomery, C.D. *Design and Analysis of Experiments.* John Wiley & Sons., 1991.

[Mosheiov 94] Mosheiov, Gur. Scheduling Jobs Under Simple Linear Deterioration. *Computers and Operations Research.* **21** 653-659, 1994.

[Nakamura 88] Nakamura, N. and Salvendy, G. An Experimental Study of Human Decision-making in Computer-based Scheduling of Flexible Manufacturing System. *International Journal of Production Reseach.* **26** (4):567-583, 1988.

[Ow 88] Ow, P.S. and Smith, S.F. and Thiriez, A. Reactive Plan Revision. *Proceedings of the seventh National Conference on Artificial Intelligence*, pages 77-82. St-Paul, Minnesota, 1988.

[Paraskevopoulos 91] Paraskevopoulos, S. and Karakitsos, E. and Rustem, B. Robust Capacity Planning Under Uncertainty. *Management Science.* **37** (7):787-800, July, 1991.

[Ringer 90] Ringer, M.J. Time Phased Abstractions for Combining Predictive and Reactive

Scheduling Methods. *1990 International Industrial Engineering Conference Proceedings*, pages 121-126. 1990.

[Rosenfield 87] Rosenfield, D. A Model for Predicting Frequencies of Random Events. *Management Science.* **33** (8), August, 1987.

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

[Sadeh 94] Sadeh, N. Micro-Opportunistic Scheduling: the Micro-Boss Factory Scheduler. *Intelligent Scheduling.* In Zweben, M. and Foc, M.S., Morgan Kaufmann, 1994.

[Saks 93] Saks, V. and Johnson, I. and Fox, M. Distribution Planning: A Constrained Heuristic Search Approach. *Proceedings of the DND Workshop on Knowledge-based Systems and Robotics*. 1993.

[Shafer] Shafer, G. and Pearl, J. *Readings in Uncertain Reasoning.* Morgan Kaufmann, .

[Shaw 87] Shaw, M.J. Knowledge-Based Scheduling in Flexible Manufacturing Systems. *Texas Instruments Technical Journal.* **special-issue** 1987.

[Smithson 89] Smithson, M. *Ignorance and Uncertainty.* Springer-Verlag, 1989.

[Tadepalli 90] Tadepalli, P. and Joshi, Varad. Real-time Scheduling Using Minimum Search. *1990 International Industrial Engineering Conference Proceedings*, pages 77-81. 1990.

[Tam 94] Tam, M.M.C. and Choi, D.H.L and Chung, W.W.C. and Cheng, T.C.E. and Chiu, P.P.K. A Predictive and Reactive Scheduling Tool Kit for Repetitive Manufacturing. Szelke, E. and Kerr, R.M. (editor), *Knowledge-Based Reactive Scheduling*Elsevier Science Publishers, 1994.

[Tang 90] Tang, C.S. The Impact of Uncertainty on a Production Line. *Management Science.* **36** (12):1518-1531, December, 1990.

[Wein 91] Wein, L.M. and Ou, Jihong. The Impact of Processing Time Knowledge on Dynamic Job-shop Scheduling. *Management Science.* **37** (8):1002-1014, August, 1991.

[Wu 92] Wu, B. and Seddon, J.J. and Currie, W.L. Computer-aided Dynamic Preventive Maintenance within the Manufacturing Environment. *International Journal of Production Reseach.* **30** (11):2683-2696, 1992.

[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.

Appendix

In this appendix we discuss the general nature of uncertainty. Different aspects of uncertainty are visited. An AI classification of uncertainty is introduced. A number of techniques from different disciplines for dealing with uncertainty are reviewed.

A.1 The Nature of Uncertainty

Uncertainty for a system is the uncertainty of the outcome of an operation or activity. In the world of real action, all future operations are uncertain to some extent. Uncertainty pervades life and can arise from many sources. It is present in most tasks that require intelligent behavior, such as planning, reasoning, problem solving, decision making, classification and many others. Consequently, the management of uncertainty is crucial to the development of computer based systems which can successfully execute the tasks.

The Sources of Uncertainty

Uncertainty exists wherever strict logical implication is not possible either because of uncertain *data* or uncertain *knowledge*, leading to the problems of how to represent uncertainty and how to reason about uncertainty respectively.

The Aspects of Uncertainty

The breadth and heterogeneity of the uncertainty concepts can be testified by the terms associated when referring to uncertainty: *imprecision, unspecific, vague, fuzzy, lack of confidence, unreliability, undecidability, variability, ignorance, ambiguity,* and so on.

As an illustration of how different aspects of uncertainty may coexist in a given context, consider a hypothetical manufacturing scenario in which opinion has been collected from a number of human schedulers on the appropriate way of scheduling the milling of 10 parts on milling machines (Table 24).

With the exception of A, all the other schedules contain some aspects of uncertainty relating to conflict or ignorance. Specifically: B and C are vague, B imprecise (or unspecific) and C fuzzy; D is a statement of subjective confidence; E is ambiguous; F and G are incomplete since F does not provide machine allocation and G does not specify an upper limit for processing time; H seems to be based on a default rule of questionable relevance; I is anomalous in relation to the other responses, possibly being an error; J suggests ignorance and later resolvable; while K is completely irrelevant to the specific issue.

Table 24. Aspects of Uncertainty

Schedule	Suggested	Aspects of Uncertainty
A	Mill on machine 1 for 100 minutes.	Certainty
B	Mill on machine 1 for 90-110 minutes.	imprecise
C	Mill on machine 1 for about 100 minutes	fuzzy
D	Mill on machine 1 (likely to be running) for 100 minutes.	subjective confidence
E	Mill on machine 1 or machine 2 for 100 minutes.	ambiguity
F	Mill for 100 minutes.	incomplete
G	Mill on machine 1 for at least 100 minutes.	incomplete
H	The usual processing time is 10 minutes each.	questionable relevance
I	Mill on machine 1 for 10 minutes.	possible error
J	Can not be scheduled at this moment. Will try again.	undecidability
K	Drill 100 holes.	irrelevance (inconsistency)

This set of responses from scheduler both individually and collectively represents a broad range of uncertainty aspects that may be encountered in the development or application of decision making systems. It also demonstrates that whereas some aspects of uncertainty apply to atomic propositions (vagueness, confidence and ambiguity), others such as incompleteness and inconsistency are thought in terms of sets of propositions.

A.2 An AI Classification of Uncertainty

There are different topologies of uncertainty concepts [Smithson 89] . They serve different purposes in specific situations and are domain oriented. Krause and Clark [Krause 93] [Krause 93] proposed a characterization of uncertainty (Figure 41) aimed at classifying uncertainty in terms of those aspects which are most relevant to the development of AI and ES applications.

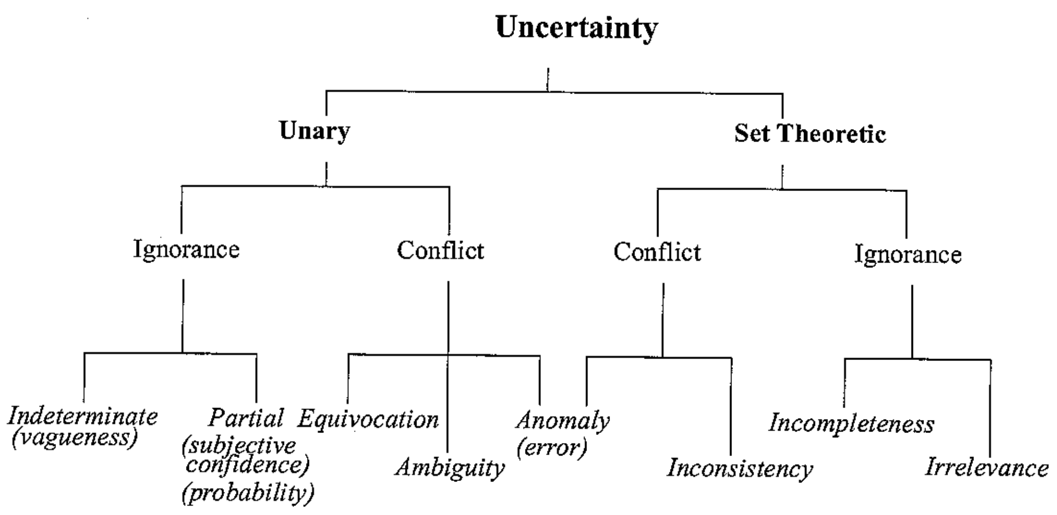


Figure 42. Uncertainty Classification for AI Systems.

At the top level, there are unary uncertainty and set theoretic uncertainty, that are applied to individual proposition and set of propositions respectively. At the middle level, the distinction is between ignorance (lack of knowledge) and conflict (conflicting knowledge). At the leaf level, further distinctions are made as following:

- *Indeterminate (or vagueness)*: reflects non-specificity and/or fuzziness of the extent of a proposition.
- *Partial* knowledge: either the subjective degree of belief (confidence) or some measure of statistical probability.
- *Equivocation*: a proposition is simultaneously both supported and discredited.
- *Ambiguity*: there are alternative non overlapping interpretations of the proposition.
- *Anomaly:* one of a set of propositions is irregular with respect to others. It can be claimed to be a property of both the set and the unary uncertainty. Error is viewed as a species of anomaly.
- *Inconsistency*: a set of propositions can not be simultaneously true.
- *Incompleteness*: some elements missing within the set of propositions.
- *Irrelevancy*: not only is the confidence in the individual proposition uncertain, but the extension of the set of propositions is also uncertain.

As the discussion in the next section on uncertainty management will show, it is the acknowledgment of aspects of uncertainty other than traditional probability (subjective or frequentistic) that has led to the development of other techniques (than Bayesian theory) for managing uncertainty in AI. The task of uncertainty management is so challenging that researchers are still striving to understand it better.

A.3 Characteristics of Uncertainty

Time Dependency

Uncertainty evolves with ***time***. As time elapses, data and knowledge are gathered to ascertain the uncertain situation. For instance, an uncertain forecast is replaced by a certain piece of data once the time progresses to the point where the real world data is observed. In forecasting, t1 can be forecasted with more precision than t2 if t1 < t2 (Figure 42). The heights of the bars at t1 and t2 represent the forecasting error ranges.

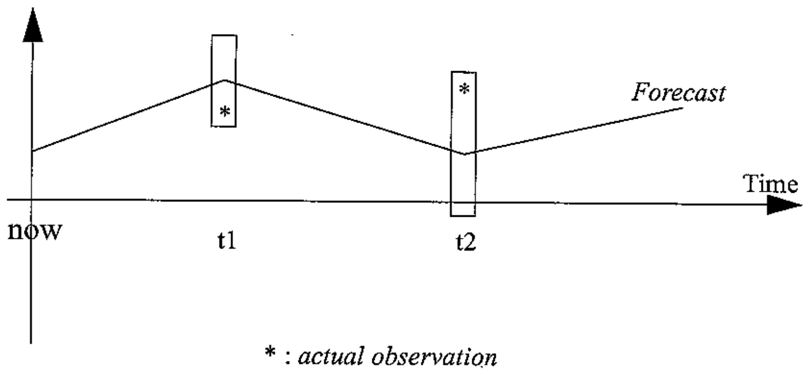


Figure 43. Forecast Precision

This leads to the idea of *delaying the time of decision* to minimize the effect of uncertainty on the decision made, i.e., delivering decisions as late as possible. A later decision bases on more facts gathered and is subject to less uncertainty than an earlier decision.

Difficulty to Quantify

Due to the variety and vagueness of uncertainty, it is difficult to parameterize a representation that can reasonably quantify human's perception of the uncertain situation. When uncertain data or uncertain knowledge is acquired from users, the inconsistency among users and within an individual may generate a wide range of different quantification for the same uncertain information.

This difficulty in uncertain data and knowledge representation challenges the mathematical reasoning of uncertain situations.

Distinguish between Subjective and Objective Uncertainty

Objective uncertainty refers to those that are statistical in nature as opposed to REAL uncertainty. The options set is known for an uncertain circumstance. Although we do not know when an option is going to happen specifically, statistical probability function can be found to define the frequency of occurrence of each option in the set. A good case of this is the coin flip-and-toss. Each side has an equal chance of appearing but the occurrence in each try is uncertain.

Subjective uncertainty bases on people's interpretation of a piece of data. It is observant oriented. One example of this is the behavioral or option uncertainty in a game of two, when your reaction bases on the behavior of your opponent. But you have no possibility of knowing all his options or a distribution function of his options. Some possible options may have been overlooked but did happened.

A.4 Uncertainty Representation and Management Techniques

Up to the 1970s, almost the only uncertainty management tool used in AI had been the probability theory. But challenges from specific fields have shown the limitations of traditional approach and new techniques were developed to respect those challenges. As Shafer and Pearl [Shafer] [Shafer 90] pointed out:

"Not all problems of uncertainty in AI lend themselves to probability. Other approaches are often required."

We are to review a few of techniques from different disciplines for dealing with uncertainty. Some general approaches will be discussed. Criteria for discussing include: What aspects of uncertain situations are especially easy or difficult to represent? Is there an approach which represents all pertinent aspects of uncertainty? How much expertise is needed to use the representation? Does the representation make any assumptions about the accuracy of the information it requires from a user? And so on. What makes representation and management of uncertainty especially interesting and extremely difficult, however, is the requirement for a formal representation that can answer the above questions favorably. On the other hand, it is a commonplace [Cohen 85] [Cohen 85] that no single representation method can adequately represent all aspects of uncertainty. Any representation language favors some concepts at the expense of others. Nevertheless, all representations expand our understanding and control of uncertainty.

Bayesian Probability

Having the longest tradition and being the best understood method for handling uncertainty, Bayesian probability theory provides a reference for other alternative formalisms. The basic expressions in the Bayesian probability theory are conditional probabilities. A mathematical axiomatization of Bayesian probability theory is as follows:

Suppose *e* being the evidence, *h* and *g* being hypotheses. Probability is a continuous monotonic function p such that: $p(h|e)$ represents the conditional probability of hypothesis h given the evidence e. i.e., probability of h being true given that e is observed to be true. This represents our degree of belief in a piece of data or knowledge.

The mathematics of probability (EQ1 to EQ4) advocate that:

- Probability values should lie in the range [0,1].
- Probability of a true hypothesis is unity.
- Either the hypothesis or its negation will be true.
- Probability of the conjunction of two hypotheses is the probability of the first hypothesis given that the second hypothesis is true, multiplied by the probability of the second hypothesis.

$$0 < p(h|e) < 1 \qquad \textbf{(EQ 16)}$$

$$p(True|e) = 1 \qquad \textbf{(EQ 17)}$$

$$p(h|e) + P(\neg h|e) = 1 \qquad \textbf{(EQ 18)}$$

$$p(gh|e) = p(h|ge) \cdot p(g|e) \qquad \textbf{(EQ 19)}$$

The power of subjective Bayesian conception lies in the constant updating of the belief in a hypothesis in response to the observation of new evidences. That is, belief is constantly revised to reflect the changing environment. This can be mathematically computed as in equation 5. That is, when a new *e* is observed, belief in *h* can be revised using its previous probability and the conditional probability of the evidence. This provides a useful tool for representing and dynamically updating the uncertainty measurement according to the real world.

$$p(h|e) = \frac{p(e|h) \cdot p(h)}{p(e)} \tag{EQ 20}$$

Certain well defined conditions have to be met while using Bayesian probability. No concepts such as inconsistency, incompleteness and irrelevance can be represented, since the theory centers on the uncertainty as it applies to atomic or conditional propositions. The complete reasoning structure is assumed to have existed, and a restructuring can be very costly since all the conditional dependencies have to be recalculated.

The Certainty Factor Model

The CF formalism is a heuristic approach to reason under uncertainty in a rule-based Expert System. It provides a method for both formalizing heuristic reasoning as deductive rules, and simultaneously allowing uncertainty to be quantified and combined within a simple calculus. It was an attempt to weaken some of the axioms of probabilistic method (replace absolute probability with relative one.) and be computational efficient. CF model has been successfully applied to medical diagnose systems such as MYCIN.

The method is not discussed in detail here. But some limitations on applying CF have to be born in mind. Basically, rules must be collectable from field experts and must reflect the inherent cause-effect structure of the problem. Non-independent evidence can not be scattered in different rules. And it is used almost solely for rule-based expert systems.

Epistemic Probability: the Dempster-Shafer Theory of Evidence

The Dempster-Shafer Theory attempted a generalization of the Bayesian model. It allows greater representational flexibility while sacrificing computational complexity.

There are at least three important distinctions between them: Firstly, the belief functions of D-S theory are set functions rather than point values, so that the belief on a group of propositions can be represented. Secondly, the absence of belief in a proposition does not necessarily imply a belief in the negation of that proposition. This allows an expression of ignorance, i.e., no commitment to the truth of a proposition or its negation. Thirdly, D-S has a rule of combination for the pooling of evidence from a variety of sources.

In terms of knowledge representation, D-S greatly extends the aspects of uncertainty that can be modeled, such as ignorance, partial knowledge, inconsistency and anomaly. The domains where D-S is a natural model of reasoning include diagnosis, classification, and so on. Hierarchy hypothesis is common to these applications and evidences come from various sources.

Fuzzy Logic - Imprecise and Vagueness

Fuzzy logic provides the formal representation and manipulation of incomplete knowledge manifested as vagueness. By allowing overlapping between the membership functions of different value levels (such as high, medium and low), some degrees of linguistic ambiguity and inconsistency can be represented. This is a rich area of research which has a clear role in the domain of uncertainty reasoning. Applications which benefit greatly from this methodology are those under which human judgement is involved or verbal ambiguity is common.

Flexibility and Diversification

In theory, if a system is flexible enough that it has prepared alternatives for all the uncertain situations, it should be less interrupted. Diversification and increasing flexibility are two ways of being prepared.

Diversification is a common practice in all industries. For instance, investors spread their investment in a broad range of areas to share the uncertainty of losing in one field. In the retailing industry, a diversified suppliers group can minimize the impact of lead time uncertainty.

An example for flexibility versus uncertainty is the staff cross-training, where employees are trained to perform multiple tasks. The uncertainty of staff demand and absenteeism is accommodated by moving multi-functional staffs around. It has become a common sense that the modern industry must be highly flexible to adapt to the changing environment. In manufacturing, flexibility is ranked behind productivity, delivery and quality in importance for organization competitiveness [Cox 89]. The following categories of FMS shop floor flexibility were categorized by Browne et al. [Browne 84] [Browne et al. 84].

- *Machine Flexibility*: the ease of making changes to produce a given set of part types.
- *Process Flexibility*: the ability to produce a set of part types in several ways. Called job flexibility by Buzacott [Buzacott 82] [Buzacott 82].
- *Product Flexibility*: the capability to change product mix rapidly and inexpensively.
- *Routing Flexibility*: the ability to continue to process jobs on alternate routes/machines in case of a breakdown.
- *Expansion Flexibility*: the ease of modularly building and expanding a system easily.
- *Operation Flexibility*: the ability to interchange the ordering of several operations that have no precedence requirements among them.

Besides the shop floor flexibility discussed above, flexibility of other components in manufacturing can be identified as well. The *flexibility of material handling system* is concerned with the ability of AGV or other transporting facility to handle different parts on a number of routes. The *computer control system flexibility* is measured by its adaptability to the changing functions. *Design flexibility* is built into the plant and product design phase. It is highly technology based and linked closely to resource flexibility.

A higher level of flexibility is the organizational flexibility that is measured by all the above flexibility. When considering increasing organizational flexibility to achieve competitiveness, there is the trade off between the economic advantage gained and the cost of the technology upgrade.

Selecting Uncertainty Management Techniques

It is agreed that there is no single universally applicable method for all uncertainty situations. Development of a new approach or selection of an existing method depends on the uncertainty aspects of the problem in concern. Some general desiderata should be respected though. Bonissone [Bonissone 87] [Bonissone 87] suggested a number of features relevant to the selection of uncertainty management techniques. The features are classified in terms of three levels (Table 25): *representation*, *inference* and *control*.

Table 25. Desiderata of Uncertainty Management

	Representation Layer
1	Representation of the amount of evidence for and against each hypothesis.
2	Representation of meta-information such as evidence source and credibility, logical dependencies, etc.
3	Allow the user to describe the uncertainty at the available levels of detail.
4	Able to detect potential conflicts and to identify factors contributing to conflict.
5	Representation of ignorance to allow non committal statements.
	Inference Layer
6	No assumptions of evidence independence.
7	The syntax and semantics should be closed under the rule of combination.
8	Propagation and summary functions should have clear semantics.
	Control Layer
9	Distinction between conflict and ignorance. Remove conflict by retracting some elements from the conflicting set. Remove ignorance by selecting a default value.
10	Aggregation and propagation of uncertainty during the reasoning must be tracable.

These factors are by no means universally accepted or complete. They highlighted a short-list of concerns in developing a system that must operate and reason under uncertainty. Moreover, it is also necessary to consider the pragmatics of system development and the run-time computational complexity when selecting uncertainty management techniques for a given situation.

A.5 Example of PODL Input Files

The scheduling problem to be solved is defined using a language: PODL. The same language is used to define strategies of solving: Policies. This is a declarative language that has a LISPish feel. The definitions stated in PODL language are translated into internal objects of ODO:TNG. The solver is constraint-based and guided by the policies. The solution is obtained in a trace file, together with other tracing information. The granularity of the tracing information is set by an environment variable: TNGTRACELEVEL.

ODO:TNG's current grammar for problem declaration and problem solving can be identified from the below example written in PODL. The extensions we made for the use of uncertainty representation and simulation are written in italic.

```
(schedule :name sched1
          :min-time 0
          :max-time 100)
(role-class :name MillingMachine :role-type Mechanism)
```

```
(role-class :name LatheMachine :role-type Mechanism)
(role-class :name PowerDrill :role-type Mechanism)

(object :name Machine1
         :has-role (role :name MillingMachine :sim-capacity 1)
         :mtbf (uniform 20 30) :fail-duration (uniform 10 15)
         :protected TP)
(object :name Machine2
         :has-role (role :name MillingMachine :sim-capacity 2))
(object :name Machine3
         :has-role (role :name LatheMachine :sim-capacity 1)
         :mtbf (uniform 20 25) :fail-duration (uniform 8 10)
         :protected TP)
(object :name Machine4
         :has-role (role :name PowerDrill :sim-capacity 1))

(activity :name a1
         :random-duration (uniform 10 20)
         :uses (object :object-name Machine1 :role-name MillingMachine
         :amount 1))
(activity :name a2 :min-duration 10 :max-duration 10
         :temporal-relation (after :activity a1 :interval 0)
         :uses (object :object-name Machine1 :role-name MillingMachine
         :amount 1))
(activity :name a3 :min-duration 10 :max-duration 10
         :temporal-relation (after :activity a2 :interval 0)
         :uses (object :object-name Machine3 :role-name LatheMachine
         :amount 1))
(activity :name a4 :min-duration 10 :max-duration 10
         :temporal-relation (after :activity a3 :interval 0)
         :uses (object :object-name Machine4 :role-name PowerDrill
         :amount 1))


(atomic-policy :name random-resource-assign
 :commitment-type assign-resource
 :forward-commitments
 (filters
   :generate (unassigned)
   :select (arbitrary)
   :score none # don't score
   :select-scored (random))
   :propagation-methods (none)
   :backtrack-method none
   :termination-criteria (cost == 0)
   :state-cost-function num-res-unassigned-activities
   :state-acceptance-criteria always)

(atomic-policy :name simple-construct
   :commitment-type assign-starttime
   :forward-commitments
 (filters :generate (unassigned smallest-pv-cardinality)
   :select (earliest)
   :score none)
```

```
    :backward-commitments
  (filters :generate (most-recent-failure))
    :propagation-methods (temporal-possible-value unit-resource)
    :backtrack-method chronological
    :termination-criteria (cost == 0 || search-iterations == 1000)
    :state-cost-function num-unassigned-activities
    :state-acceptance-criteria no-empty-pvs)

(meta-policy :name construct1
    :sub-policies (random-resource-assign simple-construct)
    :termination-criteria (cost == 0 || search-iterations == 1)
    :state-cost-function num-unassigned-activities
    :state-acceptance-criteria always)

(start-scheduling :policy construct1)
(show-activities)
(simulate)
```

A.6 Problem Solving Structure of ODO:TNG

The search strategy developed in ODO:TNG follows the commitment-oriented structure that [Davis 94] described as:

```
while (not search_termination_condition) do
   select_commitment;
   assert_commitment;
   propagate;
   if (not acceptable_resulting_state) then
      release_commitment(s);
```

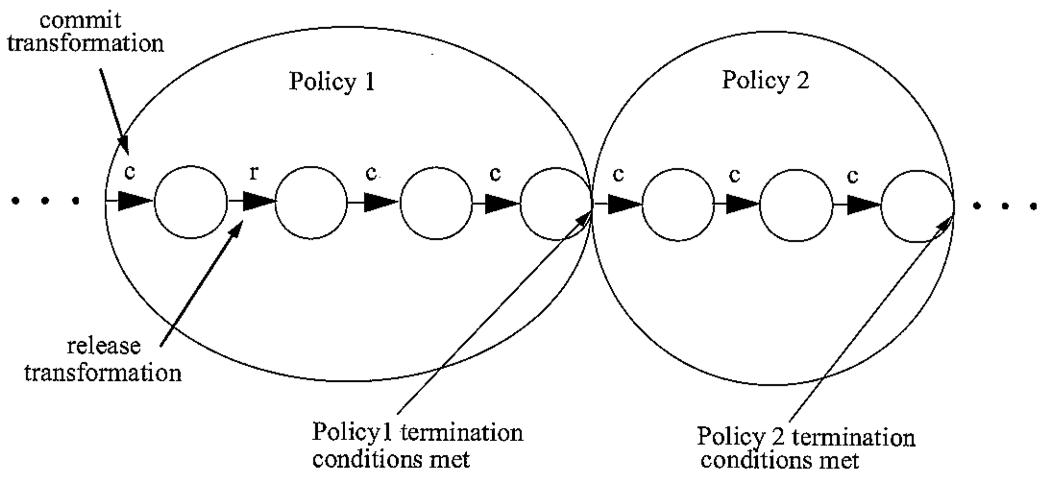

commit transformation
Policy 1
Policy 2
c
r
c
c
c
c
c
release transformation
Policy1 termination conditions met
Policy 2 termination conditions met

Figure 44. Transformations structured by multiple policies.

A policy is defined as a specification of the exact manner in which each component of the search loop is to be performed. Policies enforce structure on the commit and release transformations. A new policy can begin once a previous policy has ended. Thus we will assert that a set of commit/release transformations can be viewed as sequentially executing policies which adhere to the above decision-making model. Figure 43 shows how two or more policies can work together to perform transformational search.

ODO:TNG structures the solving process into AtomicPolicy and MetaPolicy. Each AtomicPolicy defines a set of filter-functions that arrive at a CommitmentInstance that is then asserted and propagated in the forward step of the algorithm. It also defines the backtrack mechanism, termination criteria, and other structures. The focus here is the set of filter functions that (essentially) you feed in all possible commitments (of a particular type) at the top and you get a single commitment out at the bottom.

A set of AtomicPolicies can be aggregated as sub-components of a MetaPolicy. The MetaPolicy simply calls each AtomicPolicy sequentially and then exits. There are plans to redesign the policy structure so that it can make more than one type of commitments at each state. Also needed is the ability to dynamically select the AtomicPolicy to be executed.

A.7 Software Development and Running Environment

ODO:TNG is implemented in C++. Parsing of the input commands is performed using the LEX and YACC utilities. ODO:TNG has been compiled and tested in DEC C++ compiler (cxx) in the UNIX environment on a Digital DECstation 5000/133. ODO:TNG currently consists of approximately 30,000 lines of source code. This code compiles into an executable of approximately 20 MB.

A.8 PODL input files for Experiment Problem Set

In Chapter 7, six base problems were chosen to be tested under different uncertainty scenarios. Their respective PODL problem description files are listed here. Resource Object entry is varied to input different failure descriptions. Note that the line initiated by '#' is interpreted as comments.

e0ddr1 - tight due date, 1 bottleneck

```
#############################################
# e0ddr1-10-by-5-1.tng (Norman's problem)
#
# converted to PODL by Hong
#############################################

#10 jobs, 5 operations per job
#5 machines

(schedule :name e0ddr1-fail1 :min-time 0 :max-time 199)

###########
#Resources
###########
(role-class :name machine :role-type Mechanism)

(object :name Resource0 :has-role (role :name machine :sim-capacity 1))
(object :name Resource1 :has-role (role :name machine :sim-capacity 1))
(object :name Resource2 :has-role (role :name machine :sim-capacity 1))
(object :name Resource3 :has-role (role :name machine :sim-capacity 1))
(object :name Resource4 :has-role (role :name machine :sim-capacity 1))

################
#Activities (jobs)
################

#job0
############

(activity :name A00 :duration 8
 :uses (object :object-name Resource4 :role-name machine :amount 1))

(activity :name A01 :duration 10
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A00))

(activity :name A02 :duration 13
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A01))

(activity :name A03 :duration 8
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A02))

(activity :name A04 :duration 4
 :duedate 149
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A03))


#job1
##########

(activity :name A10 :duration 10
 :uses (object :object-name Resource4 :role-name machine :amount 1))

(activity :name A11 :duration 8
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A10))

(activity :name A12 :duration 14
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A11))

(activity :name A13 :duration 5
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A12))

(activity :name A14 :duration 7
 :duedate 149
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A13))

#job2
##########

(activity :name A20 :duration 11
 :uses (object :object-name Resource3 :role-name machine :amount 1))

(activity :name A21 :duration 5
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A20))

(activity :name A22 :duration 13
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A21))

(activity :name A23 :duration 10
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A22))

(activity :name A24 :duration 8
 :duedate 149
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A23))

#job3
##########

(activity :name A30 :duration 9
 :uses (object :object-name Resource1 :role-name machine :amount 1))

(activity :name A31 :duration 7
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A30))

(activity :name A32 :duration 15
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A31))

(activity :name A33 :duration 8
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A32))

(activity :name A34 :duration 11
 :duedate 149
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A33))
```

```
#job4
##########

(activity :name A40 :duration 3
 :uses (object :object-name Resource0 :role-name machine :amount 1))

(activity :name A41 :duration 6
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A40))

(activity :name A42 :duration 11
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A41))

(activity :name A43 :duration 7
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A42))

(activity :name A44 :duration 8
 :duedate 149
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A43))

#job5
##########

(activity :name A50 :duration 8
 :uses (object :object-name Resource3 :role-name machine :amount 1))

(activity :name A51 :duration 11
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A50))

(activity :name A52 :duration 10
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A51))

(activity :name A53 :duration 9
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A52))

(activity :name A54 :duration 7
 :duedate 149
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A53))

#job6
##########

(activity :name A60 :duration 3
 :uses (object :object-name Resource3 :role-name machine :amount 1))

(activity :name A61 :duration 3
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A60))

(activity :name A62 :duration 12
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A61))

(activity :name A63 :duration 10
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A62))

(activity :name A64 :duration 9
 :duedate 149
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A63))

#job7
##########

(activity :name A70 :duration 7
 :uses (object :object-name Resource0 :role-name machine :amount 1))

(activity :name A71 :duration 11
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A70))

(activity :name A72 :duration 12
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A71))

(activity :name A73 :duration 11
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A72))

(activity :name A74 :duration 4
 :duedate 149
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A73))

#job8
##########

(activity :name A80 :duration 5
 :uses (object :object-name Resource4 :role-name machine :amount 1))

(activity :name A81 :duration 7
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A80))

(activity :name A82 :duration 10
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A81))

(activity :name A83 :duration 10
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A82))

(activity :name A84 :duration 10
 :duedate 149
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A83))

#job9
##########

(activity :name A90 :duration 7
 :uses (object :object-name Resource1 :role-name machine :amount 1))

(activity :name A91 :duration 5
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A90))

(activity :name A92 :duration 13
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A91))

(activity :name A93 :duration 3
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A92))

(activity :name A94 :duration 8
 :duedate 149
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A93))
```

e0ddr2 - tight due date,2 bottlenecks

```
##############################################
# e0ddr2-10-by-5-1.tng (Norman's problem)
##############################################
(schedule :name e0ddr2-fail1 :min-time 0 :max-time 299)

#10 jobs, 5 operations per job
#5 machines

###########
#Resources
###########
(role-class :name machine :role-type Mechanism)

(object :name Resource0 :has-role (role :name machine :sim-capacity 1))
(object :name Resource1 :has-role (role :name machine :sim-capacity 1))
(object :name Resource2 :has-role (role :name machine :sim-capacity 1)
   :mtbf (uniform 10 15) :fail-duration (uniform 2 4)
   :protected TP1)
(object :name Resource3 :has-role (role :name machine :sim-capacity 1))
(object :name Resource4 :has-role (role :name machine :sim-capacity 1)
   :mtbf (uniform 10 15) :fail-duration (uniform 2 4)
   :protected TP1)

################
#Activities (jobs)
################

#job0
############

(activity :name A00 :duration 7
 :uses (object :object-name Resource3 :role-name machine :amount 1))

(activity :name A01 :duration 8
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A00))

(activity :name A02 :duration 15
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A01))

(activity :name A03 :duration 8
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A02))
```

```
(activity :name A04 :duration 13
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A03))

#job1
##########

(activity :name A10 :duration 4
 :uses (object :object-name Resource0 :role-name machine :amount 1))

(activity :name A11 :duration 10
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A10))

(activity :name A12 :duration 15
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A11))

(activity :name A13 :duration 8
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A12))

(activity :name A14 :duration 14
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A13))

#job2
##########

(activity :name A20 :duration 11
 :uses (object :object-name Resource3 :role-name machine :amount 1))

(activity :name A21 :duration 10
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A20))

(activity :name A22 :duration 16
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A21))

(activity :name A23 :duration 11
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A22))

(activity :name A24 :duration 10
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A23))

#job3
##########

(activity :name A30 :duration 7
 :uses (object :object-name Resource1 :role-name machine :amount 1))

(activity :name A31 :duration 5
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A30))

(activity :name A32 :duration 8
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A31))

(activity :name A33 :duration 8
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A32))

(activity :name A34 :duration 14
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A33))


#job4
##########

(activity :name A40 :duration 11
 :uses (object :object-name Resource0 :role-name machine :amount 1))

(activity :name A41 :duration 11
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A40))

(activity :name A42 :duration 16
 :uses (object :object-name Resource2 :role-name machine :amount 1)

(activity :name A43 :duration 4
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A42))

(activity :name A44 :duration 10
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A43))

#job5
##########

(activity :name A50 :duration 11
 :uses (object :object-name Resource1 :role-name machine :amount 1))

(activity :name A51 :duration 7
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A50))

(activity :name A52 :duration 11
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A51))

(activity :name A53 :duration 6
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A52))

(activity :name A54 :duration 16
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A53))

#job6
##########

(activity :name A60 :duration 7
 :uses (object :object-name Resource3 :role-name machine :amount 1))

(activity :name A61 :duration 9
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A60))

(activity :name A62 :duration 13
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A61))
```

```
(activity :name A63 :duration 7
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A62))

(activity :name A64 :duration 14
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A63))

#job7
##########
(activity :name A70 :duration 9
 :uses (object :object-name Resource3 :role-name machine :amount 1))

(activity :name A71 :duration 3
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A70))

(activity :name A72 :duration 9
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A71))

(activity :name A73 :duration 7
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A72))

(activity :name A74 :duration 16
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A73))

#job8
##########

(activity :name A80 :duration 11
 :uses (object :object-name Resource1 :role-name machine :amount 1))

(activity :name A81 :duration 4
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A80))

(activity :name A82 :duration 10
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A81))

(activity :name A83 :duration 5
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A82))

(activity :name A84 :duration 13
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A83))

#job9
##########

(activity :name A90 :duration 5
 :uses (object :object-name Resource0 :role-name machine :amount 1))

(activity :name A91 :duration 10
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A90))

(activity :name A92 :duration 15
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A91))
```

```
(activity :name A93 :duration 6
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A92))

(activity :name A94 :duration 9
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A93))
```

enddr1: narrow due date, 1 bottleneck.

```
###############################################
# enddr1-10-by-5-1.tng (Norman's problem)
#
# converted to PODL by Sanket
###############################################

(schedule :name enddr1-fail1 :min-time 0 :max-time 229)

#10 jobs, 5 operations per job
#5 machines

###########
#Resources
###########
(role-class :name machine :role-type Mechanism)

(object :name Resource0 :has-role (role :name machine :sim-capacity 1))
(object :name Resource1 :has-role (role :name machine :sim-capacity 1))
(object :name Resource2 :has-role (role :name machine :sim-capacity 1)
 :mtbf (uniform 10 15) :fail-duration (uniform 2 4)
 :protected Y)

(object :name Resource3 :has-role (role :name machine :sim-capacity 1))
(object :name Resource4 :has-role (role :name machine :sim-capacity 1))

################
#Activities (jobs)
################

#job0
############

(activity :name A00 :duration 8 :min-start 9
 :uses (object :object-name Resource4 :role-name machine :amount 1))

(activity :name A01 :duration 10
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A00))

(activity :name A02 :duration 13
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A01))

(activity :name A03 :duration 8
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A02))

(activity :name A04 :duration 4
 :duedate 147
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A03))

#job1
##########

(activity :name A10 :duration 10 :min-start 5
 :uses (object :object-name Resource4 :role-name machine :amount 1))

(activity :name A11 :duration 8
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A10))

(activity :name A12 :duration 14
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A11))

(activity :name A13 :duration 5
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A12))

(activity :name A14 :duration 7 :duedate 148
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A13))

#job2
##########

(activity :name A20 :duration 11 :min-start 9
 :uses (object :object-name Resource3 :role-name machine :amount 1))

(activity :name A21 :duration 5
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A20))

(activity :name A22 :duration 13
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A21))

(activity :name A23 :duration 10
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A22))

(activity :name A24 :duration 8 :duedate 159
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A23))

#job3
##########

(activity :name A30 :duration 9 :min-start 2
 :uses (object :object-name Resource1 :role-name machine :amount 1))

(activity :name A31 :duration 7
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A30))

(activity :name A32 :duration 15
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A31))

(activity :name A33 :duration 8
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A32))
```

```
(activity :name A34 :duration 11 :duedate 156
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A33))


#job4
##########

(activity :name A40 :duration 3 :min-start 10
 :uses (object :object-name Resource0 :role-name machine :amount 1))

(activity :name A41 :duration 6
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A40))

(activity :name A42 :duration 11
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A41))

(activity :name A43 :duration 7
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A42))

(activity :name A44 :duration 8 :duedate 162
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A43))

#job5
##########

(activity :name A50 :duration 8 :min-start 7
 :uses (object :object-name Resource3 :role-name machine :amount 1))

(activity :name A51 :duration 11
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A50))

(activity :name A52 :duration 10
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A51))

(activity :name A53 :duration 9
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A52))

(activity :name A54 :duration 7 :duedate 160
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A53))

#job6
##########

(activity :name A60 :duration 3 :min-start 0
 :uses (object :object-name Resource3 :role-name machine :amount 1))

(activity :name A61 :duration 3
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A60))

(activity :name A62 :duration 12
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A61))
```

```
(activity :name A63 :duration 10
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A62))

(activity :name A64 :duration 9 :duedate 149
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A63))

#job7
##########

(activity :name A70 :duration 7 :min-start 4
 :uses (object :object-name Resource0 :role-name machine :amount 1))

(activity :name A71 :duration 11
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A70))

(activity :name A72 :duration 12
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A71))

(activity :name A73 :duration 11
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A72))

(activity :name A74 :duration 4 :duedate 152
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A73))

#job8
##########

(activity :name A80 :duration 5 :min-start 6
 :uses (object :object-name Resource4 :role-name machine :amount 1))

(activity :name A81 :duration 7
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A80))

(activity :name A82 :duration 10
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A81))

(activity :name A83 :duration 10
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A82))

(activity :name A84 :duration 10 :duedate 149
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A83))

#job9
##########

(activity :name A90 :duration 7 :min-start 11
 :uses (object :object-name Resource1 :role-name machine :amount 1))

(activity :name A91 :duration 5
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A90))

(activity :name A92 :duration 13
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A91))

(activity :name A93 :duration 3
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A81))

(activity :name A83 :duration 10
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A82))

(activity :name A84 :duration 10 :duedate 149
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A83))

#job9
##########

(activity :name A90 :duration 7 :min-start 11
 :uses (object :object-name Resource1 :role-name machine :amount 1))

(activity :name A91 :duration 5
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A90))

(activity :name A92 :duration 13
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A91))

(activity :name A93 :duration 3
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A92))

(activity :name A94 :duration 8 :duedate 159
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A93))
```

enddr2 - narrow due date,2 bottlenecks

```
##############################################
# enddr2-10-by-5-1.tng (Norman's problem)
#
# converted to PODL by Hong
##############################################

(schedule :name enddr2-fail1 :min-time 0 :max-time 299)

#10 jobs, 5 operations per job
#5 machines

###########
#Resources
###########
(role-class :name machine :role-type Mechanism)

(object :name Resource0 :has-role (role :name machine :sim-capacity 1))
(object :name Resource1 :has-role (role :name machine :sim-capacity 1))
(object :name Resource2 :has-role (role :name machine :sim-capacity 1)
 :mtbf (uniform 10 15) :fail-duration (uniform 2 4)
 :protected Y)
(object :name Resource3 :has-role (role :name machine :sim-capacity 1))
```

```
(object :name Resource4 :has-role (role :name machine :sim-capacity 1)
 :mtbf (uniform 30 40) :fail-duration (uniform 7 11)
 :protected Y)

################
#Activities (jobs)
################

#job0
############

(activity :name A00 :duration 7 :min-start 3
 :uses (object :object-name Resource3 :role-name machine :amount 1))

(activity :name A01 :duration 8
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A00))

(activity :name A02 :duration 15
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A01))

(activity :name A03 :duration 8
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A02))

(activity :name A04 :duration 13 :duedate 176
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A03))


#job1
##########

(activity :name A10 :duration 4 :min-start 4
 :uses (object :object-name Resource0 :role-name machine :amount 1))

(activity :name A11 :duration 10
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A10))

(activity :name A12 :duration 15
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A11))

(activity :name A13 :duration 8
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A12))

(activity :name A14 :duration 14 :duedate 174
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A13))

#job2
##########

(activity :name A20 :duration 11 :min-start 11
 :uses (object :object-name Resource3 :role-name machine :amount 1))

(activity :name A21 :duration 10
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A20))

(activity :name A22 :duration 16
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A21))

(activity :name A23 :duration 11
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A22))

(activity :name A24 :duration 10 :duedate 181
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A23))

#job3
##########

(activity :name A30 :duration 7 :min-start 9
 :uses (object :object-name Resource1 :role-name machine :amount 1))

(activity :name A31 :duration 5
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A30))

(activity :name A32 :duration 8
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A31))

(activity :name A33 :duration 8
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A32))

(activity :name A34 :duration 14 :duedate 180
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A33))


#job4
##########

(activity :name A40 :duration 11 :min-start 0
 :uses (object :object-name Resource0 :role-name machine :amount 1))

(activity :name A41 :duration 11
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A40))

(activity :name A42 :duration 16
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A41))

(activity :name A43 :duration 4
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A42))

(activity :name A44 :duration 10 :duedate 172
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A43))

#job5
##########

(activity :name A50 :duration 11 :min-start 7
 :uses (object :object-name Resource1 :role-name machine :amount 1))

(activity :name A51 :duration 7
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A50))

(activity :name A52 :duration 11
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A51))

(activity :name A53 :duration 6
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A52))

(activity :name A54 :duration 16 :duedate 177
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A53))

#job6
##########

(activity :name A60 :duration 7 :min-start 0
 :uses (object :object-name Resource3 :role-name machine :amount 1))

(activity :name A61 :duration 9
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A60))

(activity :name A62 :duration 13
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A61))

(activity :name A63 :duration 7
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A62))

(activity :name A64 :duration 14 :duedate 174
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A63))

#job7
##########

(activity :name A70 :duration 9 :min-start 1
 :uses (object :object-name Resource3 :role-name machine :amount 1))

(activity :name A71 :duration 3
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A70))

(activity :name A72 :duration 9
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A71))

(activity :name A73 :duration 7
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A72))

(activity :name A74 :duration 16 :duedate 168
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A73))

#job8
##########

(activity :name A80 :duration 11 :min-start 4
 :uses (object :object-name Resource1 :role-name machine :amount 1))
```

```
(activity :name A81 :duration 4
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A80))

(activity :name A82 :duration 10
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A81))

(activity :name A83 :duration 5
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A82))

(activity :name A84 :duration 13 :duedate 179
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A83))

#job9
##########

(activity :name A90 :duration 5 :min-start 4
 :uses (object :object-name Resource0 :role-name machine :amount 1))

(activity :name A91 :duration 10
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A90))

(activity :name A92 :duration 15
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A91))

(activity :name A93 :duration 6
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A92))

(activity :name A94 :duration 9 :duedate 179
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A93))
```

ewddr1 - wide due date, 1 bottleneck

```
################################################
# ewddr1-10-by-5-1.tng (Norman's problem)
#
# converted to PODL by Sanket
################################################

(schedule :name ewddr1-fail1 :min-time 0 :max-time 299)

#10 jobs, 5 operations per job
#5 machines

###########
#Resources
###########
(role-class :name machine :role-type Mechanism)

(object :name Resource0 :has-role (role :name machine :sim-capacity 1))
(object :name Resource1 :has-role (role :name machine :sim-capacity 1))
(object :name Resource2 :has-role (role :name machine :sim-capacity 1)
 :mtbf (uniform 10 15) :fail-duration (uniform 2 4)
 :protected Y)
(object :name Resource3 :has-role (role :name machine :sim-capacity 1))
(object :name Resource4 :has-role (role :name machine :sim-capacity 1))

################
#Activities (jobs)
################

#job0
############

(activity :name A00 :duration 8 :min-start 19
 :uses (object :object-name Resource4 :role-name machine :amount 1))

(activity :name A01 :duration 10
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A00))

(activity :name A02 :duration 13
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A01))

(activity :name A03 :duration 8
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A02))

(activity :name A04 :duration 4 :duedate 143
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A03))


#job1
##########

(activity :name A10 :duration 10 :min-start 12
 :uses (object :object-name Resource4 :role-name machine :amount 1))

(activity :name A11 :duration 8
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A10))

(activity :name A12 :duration 14
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A11))

(activity :name A13 :duration 5
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A12))

(activity :name A14 :duration 7 :duedate 144
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A13))

#job2
##########

(activity :name A20 :duration 11 :min-start 21
 :uses (object :object-name Resource3 :role-name machine :amount 1))

(activity :name A21 :duration 5
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A20))

(activity :name A22 :duration 13
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A21))

(activity :name A23 :duration 10
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A22))

(activity :name A24 :duration 8 :duedate 170
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A23))

#job3
##########

(activity :name A30 :duration 9 :min-start 5
 :uses (object :object-name Resource1 :role-name machine :amount 1))

(activity :name A31 :duration 7
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A30))

(activity :name A32 :duration 15
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A31))

(activity :name A33 :duration 8
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A32))

(activity :name A34 :duration 11 :duedate 161
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A33))

#job4
##########

(activity :name A40 :duration 3 :min-start 22
 :uses (object :object-name Resource0 :role-name machine :amount 1))

(activity :name A41 :duration 6
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A40))

(activity :name A42 :duration 11
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A41))

(activity :name A43 :duration 7
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A42))

(activity :name A44 :duration 8 :duedate 175
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A43))

#job5
##########

(activity :name A50 :duration 8 :min-start 17
 :uses (object :object-name Resource3 :role-name machine :amount 1))

(activity :name A51 :duration 11
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A50))
 (activity :name A52 :duration 10
  :uses (object :object-name Resource2 :role-name machine :amount 1)
  :temporal-relation (after :activity A51))

 (activity :name A53 :duration 9
  :uses (object :object-name Resource0 :role-name machine :amount 1)
  :temporal-relation (after :activity A52))

 (activity :name A54 :duration 7 :duedate 170
  :uses (object :object-name Resource4 :role-name machine :amount 1)
  :temporal-relation (after :activity A53))

#job6
##########

 (activity :name A60 :duration 3 :min-start 0
  :uses (object :object-name Resource3 :role-name machine :amount 1))

 (activity :name A61 :duration 3
  :uses (object :object-name Resource0 :role-name machine :amount 1)
  :temporal-relation (after :activity A60))

 (activity :name A62 :duration 12
  :uses (object :object-name Resource2 :role-name machine :amount 1)
  :temporal-relation (after :activity A61))

 (activity :name A63 :duration 10
  :uses (object :object-name Resource4 :role-name machine :amount 1)
  :temporal-relation (after :activity A62))

 (activity :name A64 :duration 9 :duedate 147
  :uses (object :object-name Resource1 :role-name machine :amount 1)
  :temporal-relation (after :activity A63))

#job7
##########

 (activity :name A70 :duration 7 :min-start 10
  :uses (object :object-name Resource0 :role-name machine :amount 1))

 (activity :name A71 :duration 11
  :uses (object :object-name Resource4 :role-name machine :amount 1)
  :temporal-relation (after :activity A70))

 (activity :name A72 :duration 12
  :uses (object :object-name Resource2 :role-name machine :amount 1)
  :temporal-relation (after :activity A71))

 (activity :name A73 :duration 11
  :uses (object :object-name Resource3 :role-name machine :amount 1)
  :temporal-relation (after :activity A72))

 (activity :name A74 :duration 4 :duedate 154
  :uses (object :object-name Resource1 :role-name machine :amount 1)
  :temporal-relation (after :activity A73))

#job8
##########

 (activity :name A80 :duration 5 :min-start 15
  :uses (object :object-name Resource4 :role-name machine :amount 1))

 (activity :name A81 :duration 7
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A80))

(activity :name A82 :duration 10
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A81))

(activity :name A83 :duration 10
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A82))

(activity :name A84 :duration 10 :duedate 147
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A83))

#job9
##########

(activity :name A90 :duration 7 :min-start 25
 :uses (object :object-name Resource1 :role-name machine :amount 1))

(activity :name A91 :duration 5
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A90))

(activity :name A92 :duration 13
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A91))

(activity :name A93 :duration 3
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A92))

(activity :name A94 :duration 8 :duedate 162
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A93))
```

ewddr2 - wide due date,2 bottlenecks

```
###############################################
# ewddr2-10-by-5-1.tng (Norman's problem)
#
# converted to PODL by Sanket
###############################################

(schedule :name ewddr2-fail1 :min-time 0 :max-time 299)

#10 jobs, 5 operations per job
#5 machines

###########
#Resources
###########
(role-class :name machine :role-type Mechanism)

(object :name Resource0 :has-role (role :name machine :sim-capacity 1))
(object :name Resource1 :has-role (role :name machine :sim-capacity 1))
(object :name Resource2 :has-role (role :name machine :sim-capacity 1)
 :mtbf (uniform 10 15) :fail-duration (uniform 2 4)
 :protected Y)
```

```
(object :name Resource3 :has-role (role :name machine :sim-capacity 1))
(object :name Resource4 :has-role (role :name machine :sim-capacity 1)
 :mtbf (uniform 30 40) :fail-duration (uniform 7 11)
 :protected Y)


#################
#Activities (jobs)
#################

#job0
#############

(activity :name A00 :duration 7 :min-start 6
 :uses (object :object-name Resource3 :role-name machine :amount 1))

(activity :name A01 :duration 8
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A00))

(activity :name A02 :duration 15
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A01))

(activity :name A03 :duration 8
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A02))

(activity :name A04 :duration 13 :duedate 182
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A03))


#job1
##########

(activity :name A10 :duration 4 :min-start 8
 :uses (object :object-name Resource0 :role-name machine :amount 1))

(activity :name A11 :duration 10
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A10))

(activity :name A12 :duration 15
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A11))

(activity :name A13 :duration 8
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A12))

(activity :name A14 :duration 14 :duedate 179
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A13))

#job2
##########

(activity :name A20 :duration 11 :min-start 25
 :uses (object :object-name Resource3 :role-name machine :amount 1))

(activity :name A21 :duration 10
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A20))
```

```
(activity :name A22 :duration 16
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A21))

(activity :name A23 :duration 11
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A22))

(activity :name A24 :duration 10 :duedate 192
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A23))

#job3
##########

(activity :name A30 :duration 7 :min-start 21
 :uses (object :object-name Resource1 :role-name machine :amount 1))

(activity :name A31 :duration 5
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A30))

(activity :name A32 :duration 8
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A31))

(activity :name A33 :duration 8
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A32))

(activity :name A34 :duration 14 :duedate 191
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A33))


#job4
##########

(activity :name A40 :duration 11 :min-start 0
 :uses (object :object-name Resource0 :role-name machine :amount 1))

(activity :name A41 :duration 11
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A40))

(activity :name A42 :duration 16
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A41))

(activity :name A43 :duration 4
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A42))

(activity :name A44 :duration 10 :duedate 174
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A43))

#job5
##########

(activity :name A50 :duration 11 :min-start 16
 :uses (object :object-name Resource1 :role-name machine :amount 1))
```

```
(activity :name A51 :duration 7
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A50))

(activity :name A52 :duration 11
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A51))

(activity :name A53 :duration 6
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A52))

(activity :name A54 :duration 16 :duedate 184
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A53))

#job6
##########

(activity :name A60 :duration 7 :min-start 0
 :uses (object :object-name Resource3 :role-name machine :amount 1))

(activity :name A61 :duration 9
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A60))

(activity :name A62 :duration 13
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A61))

(activity :name A63 :duration 7
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A62))

(activity :name A64 :duration 14 :duedate 178
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A63))

#job7
##########

(activity :name A70 :duration 9 :min-start 4
 :uses (object :object-name Resource3 :role-name machine :amount 1))

(activity :name A71 :duration 3
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A70))

(activity :name A72 :duration 9
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A71))

(activity :name A73 :duration 7
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A72))

(activity :name A74 :duration 16 :duedate 165
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A73))

#job8
##########

(activity :name A80 :duration 11 :min-start 9
 :uses (object :object-name Resource1 :role-name machine :amount 1))

(activity :name A81 :duration 4
 :uses (object :object-name Resource0 :role-name machine :amount 1)
 :temporal-relation (after :activity A80))

(activity :name A82 :duration 10
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A81))

(activity :name A83 :duration 5
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A82))

(activity :name A84 :duration 13 :duedate 189
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A83))

#job9
##########

(activity :name A90 :duration 5 :min-start 10
 :uses (object :object-name Resource0 :role-name machine :amount 1))

(activity :name A91 :duration 10
 :uses (object :object-name Resource3 :role-name machine :amount 1)
 :temporal-relation (after :activity A90))

(activity :name A92 :duration 15
 :uses (object :object-name Resource2 :role-name machine :amount 1)
 :temporal-relation (after :activity A91))

(activity :name A93 :duration 6
 :uses (object :object-name Resource1 :role-name machine :amount 1)
 :temporal-relation (after :activity A92))

(activity :name A94 :duration 9 :duedate 187
 :uses (object :object-name Resource4 :role-name machine :amount 1)
 :temporal-relation (after :activity A93))
```