To appear in: Intelligent Scheduling, edited by Zweben/Fox; Publisher Morgan Kaufmann, CA (1994)

ISIS: A Retrospective

Mark S. Fox

Department of Industrial Engineering
University of Toronto
4 Taddle Creek Road
Toronto, Ontari M5S 1A4

tel:1-416-978-6823 fax:1-416-9711373 internet:msf@ie.utoronto.ca

1.0 Introduction

Since the 1960s, planning has captured the interest of many AI researchers. *Planning selects and sequences activities such that they achieve one or more goals and satisfy a set of domain constraints.* For the most part, planning research has focused on finding a feasible chain of actions that accomplish one or more goals. Even though Simon pointed out in 1972 the problem of allocating resources over time [Simon 72], it was not until the early 1980s that scheduling came under serious scrutiny and, more recently, has garnered the attention of a significant number of AI researchers, primarily in the domains of manufacturing, military transportation, and space[1]. *Scheduling selects among alternative plans, and assigns resources and times for each activity so that the assignments obey the temporal restrictions of activities and the capacity limitations of a set of shared resources.* Scheduling is an optimization task where limited resources are allocated over time among both parallel and sequential activities such that measures like tardiness, work-in-process inventory, and makespan are minimized. Both problems have been proven to be NP-Hard [Chapman 87] [Garey & Johnson 79].

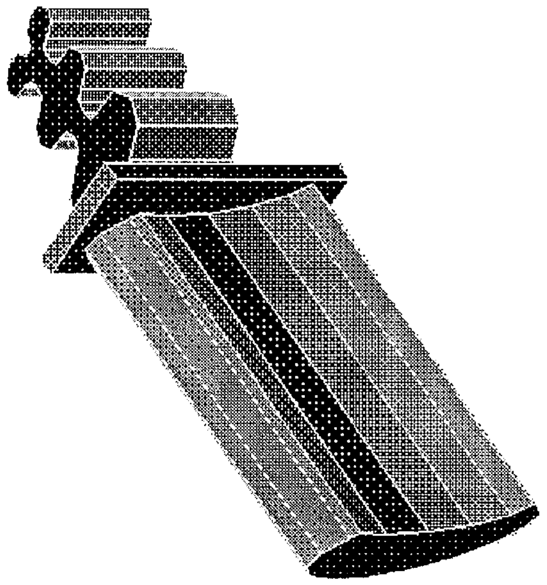
Thirteen years ago, the "real world" foisted our group at Carnegie Mellon University into the midst of the scheduling problem. The work that followed ultimately led to the creation of the ISIS system. This paper reviews the ISIS series of systems, examines the ideas those systems introduced, and reflects upon their relevance today.

2.0 Problem Definition

In 1980, we were asked to explore the application of AI techniques to a turbine component plant's job-shop scheduling problem. The primary product of the plant was steam turbine blades. A turbine blade is a complex three dimensional object produced by a sequence of forging, milling, grinding and finishing operations to tolerances of a thousandth of an inch. Thousands of different

1. There is even a conference on the topic, "Expert Systems in Production Operations and Management", and workshops on the topic have been held at AAAI.

styles of blades were produced in the plant; many were used as replacements in turbines in service.



The plant continuously received orders for one to a thousand blades at a time. Orders fell into at least six categories:

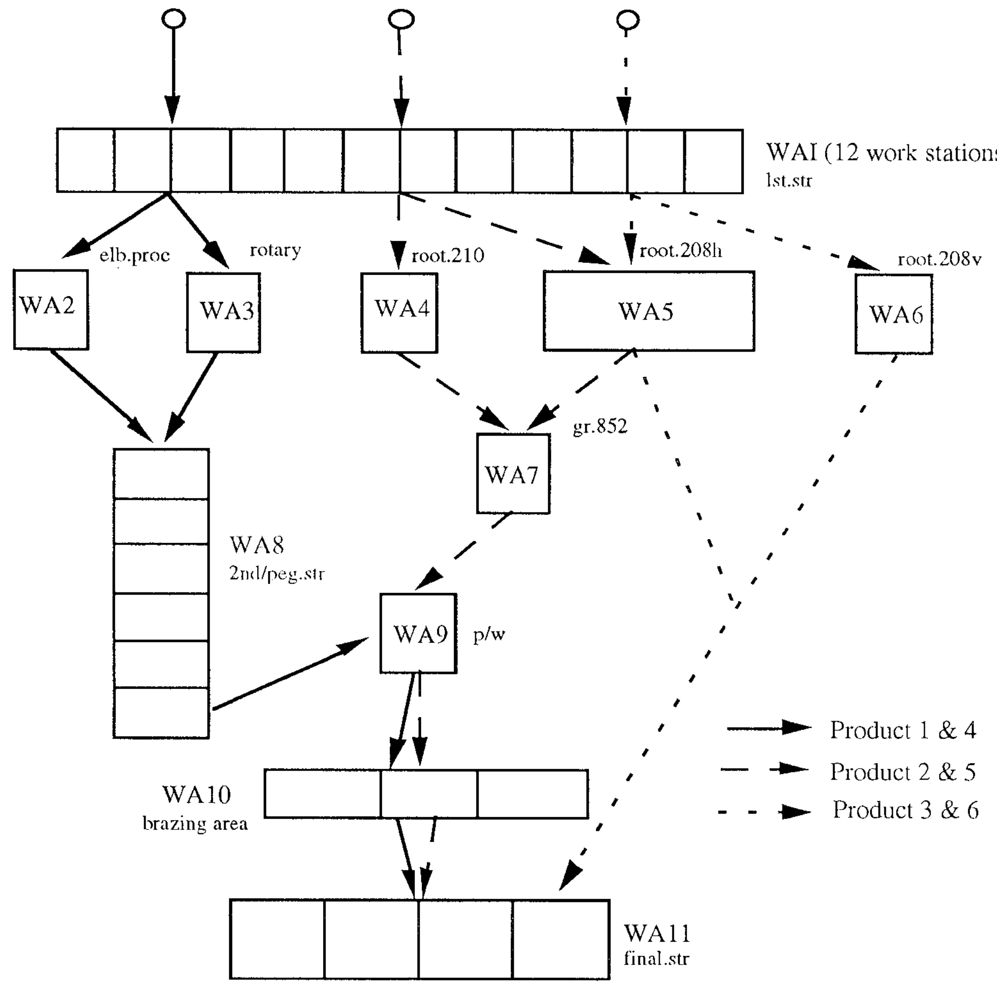
- **Forced outages:** Orders to replace blades which malfunctioned during operation. It is important to ship these orders as soon as possible, no matter what the cost.
- **Critical replacement and Ship Direct:** Orders to replace blades during scheduled maintenance. Advance warning is provided, but the blades must arrive on time.
- **Service and shop orders:** Orders for new turbines. Lead times of up to three years may be known.
- **Stock orders:** Order for blades to be placed in stock for future needs.

The portion of the plant studied has from 100 to 200 orders in process at any time.

Parts are produced according to a process routing. A routing specifies a sequence of operations on the part. An operation is an activity which defines:

- Resources required such as tools, materials, fixtures, and machines,
- Machine setup and run times, and
- Labor requirements.

In the plant, each part number has one or more process routings containing ten or more operations[1]. Process routing variations may be as simple as substituting a different machine, or as complex as changing the manufacturing process. Furthermore, the resources needed for an operation may also be needed by other operations in the shop. The following flow diagram shows the path that different parts take through the factory. Many of the areas and machines are contended for by different parts.



In AI terms, job-shop scheduling is a planning problem with the following characteristics:

- It is a *time-based* planning problem (i.e., scheduling) in which activities must be selected, sequenced, and assigned resources and time of execution.
- It is a *multi-agent* planning problem. Each order represents a separate agent for which a plan/schedule is to be created. The number of agents to be scheduled is in the hundreds and somtimes thousands.
- The agents are *uncooperative*. Each is attempting to maximize its own goals.

1. Multiple process routings correspond to a network of activities, each path representing a separate plan.

- *Resource contention* is high, hence closely coupling decisions.
- Search is *combinatorially explosive.* 85 orders moving through 10 operations without alternatives, with a single substitutable machine for each operation and no machine idle time has over 10^{1000} possible schedules.

An expert systems approach was used to construct the scheduler. This approach assumed that one or more experts could be interviewed to acquire the rules which govern their decision process. During our discussions, we found that orders were not scheduled in a uniform manner. Each scheduling choice entailed side effects whose importance varied by order. One factor that continuously appeared was the reliance of the scheduler on information other than due dates, process routings, and machine availability. The types and sources of this information were found by examining the documents issued by the scheduler. A schedule was distributed to persons in each department in the plant. Each recipient could provide information which could alter the existing schedule. In support of this observation, we found that the scheduler was spending 10%-20% of the time scheduling, and 80%-90% of the time communicating with other employees to determine what additional "constraints" could affect an order's schedule. These constraints included operation precedence, operation alternatives, operation preferences, machine alternatives and preferences, tool availability, fixture availability, NC program availability, order sequencing, setup time reduction, machine breakdowns, machine capabilities, work-in-process time, due dates, start dates, shop stability, cost, quality, and personnel capabilities/availability.

From this analysis, we concluded that the object of scheduling is not only meeting due dates, but satisfying the many constraints found in various parts of the plant. Scheduling is not a distinct function, separate from the rest of the plant, but is highly connected to and dependent upon decisions being made elsewhere in the plant. The added complexity imposed by these constraints leads schedulers to produce poor schedules. Indicators such as high work-in-process, tardiness, and low machine utilization support this conclusion[1]. Hence, any solution to the job-shop scheduling problem must identify the set of scheduling constraints, and their affect on the scheduling process.

Once the issue of designing a constraint-directed scheduling system was identified, we decided to solve the problem by constructing a family of systems. Our purpose was to investigate the performance of successively more sophisticated search architectures. At each stage, experiments were run to measure the effectiveness of the architecture.

In the remaining sections, we review the evolution of ISIS, at each stage identifying the key ideas of each version. Lastly, we reflect upon the long term impact of ISIS.

1. It is unfair to measure a scheduler's preformance based on the above measures alone. Our analysis has shown that scheduling is a complex constraint satisfaction problem, where the above indicators illustrate only a subset of constraints that the scheduler must consider. Schedulers are expert in acquiring and "juggling" the satisfaction of constraints.

3.0 ISIS-0

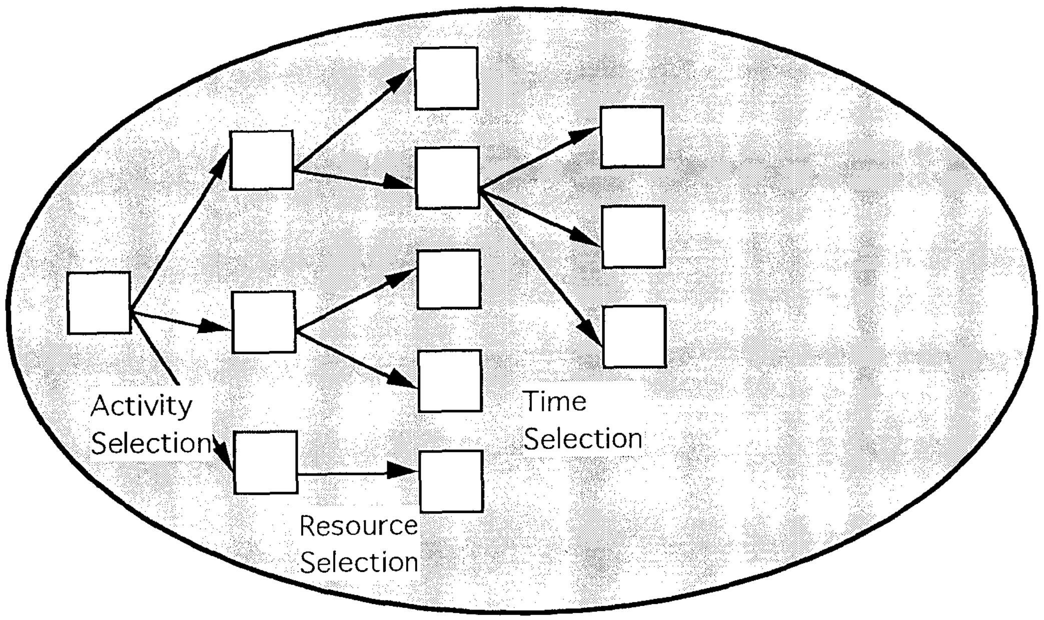
In the fall of 1980, work began on ISIS-0. Realizing that the number and variety of constraints is what makes scheduling difficult, we set out to identify the types of constraints used by schedulers, and the extent to which they could prune the space of alternative schedules using a simple search method.

3.1 Architecture

ISIS-0 employed a simple best-first, backtracking approach using constraints as a dynamically defined evaluation function. The salient points of the search architecture include:

- Each order was scheduled separately, in priority order, as determined by a combination of order category and due date.
- Search could be performed forward from the order's start date or backward from the order's due date.
- Operators would generate alternative operations, machines, and operation times. The shop was loaded hence the availability of resources at a particular time was restricted.

Incremental Generation of Partial Schedules for a Single Order



- States represent partial schedules. A path through the network determines a complete schedule.
- Constraints were either imposed exogenously by the scheduling person upon the system, or were already embedded in the factory model, and their applicability determined at each point in the search space.

- Propagation of constraints was performed when scheduling decisions early in the search path restricted decisions further on.

Research on version 0 of ISIS yielded five broad categories of constraints. The first category encountered is what we called an *Organizational Goal.* Part of the organization planning process is the generation of measures of how the organization is to perform. These measures act as constraints on one or more organization variables. An organizational goal constraint can be viewed as an expected value of some organization variable. For example:

Due Dates: A major concern of a factory is meeting due dates. The lateness of an order affects customer satisfaction.

Work-In-Process: Work-in-process (WIP) inventory levels are another concern. WIP inventory represents a substantial investment in raw materials and added value. These costs are not recoverable until delivery. Hence, reducing WIP time is desirable.

Resource Levels: Another concern is maintaining adequate levels of resources necessary to sustain operations. Resources include personnel, raw materials, tools, etc. Each resource will have associated constraints. For example, labor size must be smoothed over a month's interval, or raw materials inventory may have to be limited to a two day supply.

Costs: Cost reduction can be another important goal. Costs may include material costs, wages, and lost opportunity. Reducing costs may help achieve other goals such as stabilization of the work force.

Production Levels: Advance planning also sets production goals for each cost center in the plant. This serves two functions: it designates the primary facilities of the plant by specifying higher production goals, and also specifies a preliminary budget by predicting how much the plant will produce. One outcome of this activity is a forecast of the work shifts that will be run in various areas of the plant.

Shop Stability: Shop stability is a function of the number of revisions to a schedule and the amount of disturbance in preparation caused by these revisions. It is an artifact of the time taken to communicate change in the plant and the preparation time.

One can view all organizational goal constraints as being approximations of a simple profit constraint. The goal of an organization is to maximize profits. Scheduling decisions are then made on the basis of current and future costs incurred. For example, not meeting a due date may result in the loss of a customer and, in turn, erosion of profits. The longer the work in process time, the greater the carrying charge will be for raw materials and value-added operations. Maintaining a designated production level may distribute the cost of the capital equipment in a uniform manner. In practice, most of these costs cannot be accurately determined and must therefore be estimated.

Physical constraints determine a second category of constraint. Physical constraints specify characteristics which limit functionality. For example, the length of a milling machine's workbed may limit the types of turbine blades for which it can be used. Similarly, there are specific machine set-up and processing times associated with different manufacturing operations.

Causal restrictions constitute a third category of constraint. They define what conditions must be satisfied before initiating an operation. Examples of causal constraints include:

- **Precedence**: A process routing is a sequence of operations. A precedence constraint on an operation states that another operation must take place before (or after) it. There may be further modifiers on the constraint in terms of minimum or maximum time between operations, product temperature to be maintained, etc.
- **Resource Requirements**: Another causal constraint is the specification of resources that must be present before or during the execution of a process. For example, a milling operation requires the presence of certain tools, an operator, fixtures, etc.

A fourth category of constraint is concerned with the **availability** of resources. As resources are assigned to specific operations during the production of a schedule, constraints declaring the resources unavailable for other uses during the relevant time periods must be generated and associated with these resources. Resource availability is also constrained by the work shifts designated in the plant, machine maintenance schedules, and other machine down times (e.g. breakdowns).

A fifth category of constraint is **preference** or what has come to be known as a "soft" constraint. A preference constraint can also be viewed as an abstraction of other types of constraints. Consider a preference for a machine. It expresses a floor supervisor's desire that one machine be used instead of another. The reason for the preference may be due to cost or quality, but sufficient information does not exist to derive actual costs. In addition to machine preferences, operation preferences, and order sequencing preferences exemplify this type of constraint.

The following constraints are those we have identified, as well as the categories we have used to classify them.

TABLE 1.

Constraint	**Organization Goal**	**Physical**	**Causal**	**Preference**	**Availability**
Operation Alternatives			X		
Operation Preferences				X	
Machine Alternatives			X		
Machine Preferences				X	
Machine Physical Constraints		X			
Set-up Times	X	X			
Machine Queue Ordering Pref.				X	
Machine Queue Stability	X				
Due Date	X				
Work-in-Process	X				
Tool Requirement			X		
Material Requirement			X		
Personnel Requirement			X		
Resource Reservation					X

TABLE 1.

Constraint	Organization Goal	Physical	Causal	Preference	Availability
Shifts					X
Down Time					X
Cost	X				
Productivity Goals	X				
Quality	X	X			
Inter-Operation Transfer Time			X		

3.2 Performance

ISIS-0 was completed in December, 1980 and partially demonstrated, bugs and all, at the sponsoring plant. While it demonstrated that schedules could be constructed by generating alternative process plans and resource assignments, and pruned using constraints, its knowledge was not strong enough to deal with the complexity of the problem.

3.3 Constraint-Directed Search Concepts

Constraint-Directed Evaluation. ISIS-0 dynamically constructed a different evaluation function for each state in the search space. It constructed the evaluation function out of the constraints which were resolved to be applicable to the state under consideration. Each constraint contributed both an importance (i.e., weight) and utility. Constraints were resolved by extracting them from the resources and operations defined in the particular state.

4.0 ISIS-1: Constraint Directed Scheduling

ISIS-0 identified the broad categories of constraints but more work on the representation and search architecture was required. In January, 1981, work on the second version of ISIS began. The purpose of this system was twofold. Given the central role of constraints to determine a job shop schedule, a major focus of our research was to identify and characterize the constraint knowledge required to support an effective constraint-directed search. Consider the imposition of a due date: In its simplest form, this constraint would be represented by a date alone, implying that the job be shipped on that date. In actuality, however, due dates may not always be met, and such a representation provides no information as to how to proceed in these situations. An appropriate representation must include the additional information about the due date that may be necessary in constructing a satisfactory schedule. For example:

- How important is the constraint relative to the other known constraints? Is it more important to satisfy the cost constraint than the due date?
- If I cannot find a schedule which satisfies the constraint, are there relaxations of the constraint which can be satisfied. I.e., is there another due date which is almost as good?

- If there are relaxations available for the constraint, are any more preferred? Perhaps I would rather ship the order early rather than late.
- If I chose a particular relaxation, how will it affect the other constraints I am trying to satisfy? Will meeting the due date negatively or positively affect the cost of the order?
- Under what conditions am I obliged to satisfy a constraint? What if there are two constraints specified for the same variable, i.e., two different due date for the same lot? Or there may two different due dates depending on the time of year.

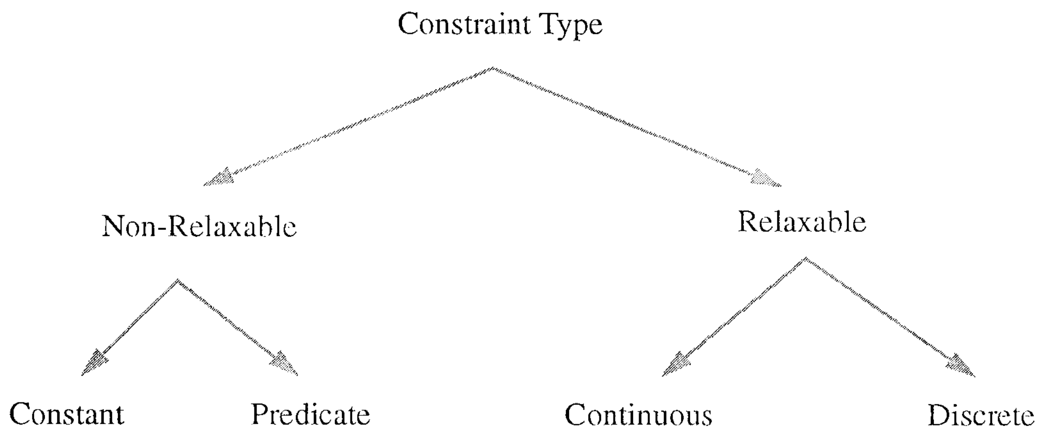
In essence, a constraint is not simply a restriction on the value of a slot for example, but the aggregation of a variety of knowledge used in the reasoning process.

The second goal was to measure the effectiveness of the a modified Beam search [Lowerre & Reddy 90] architecture which uses constraints.

4.1 Constraint Representation

Given the importance of constraints in the construction of schedules, we first examine the types of constraint knowledge needed to be represented and then give an example of one constraint: a due date.

One of the central issues that must be addressed by the constraint representation is *conflict*. Consider cost and due-date constraints. The former may require reduction of costs while the latter may require shipping the order in a short period of time. To accomplish the latter, faster, more expensive machines may be required, thereby causing a conflict with the former. In short, it may not be possible to satisfy both constraints, in which case one or both must be relaxed. This is implicitly accomplished in mathematical programming and decision theory by means of utility functions and the specifications of relaxation through bounds on a variable's value. In AI, bounds on a variable are typically specified by predicates [Brown 86] [Stefik 81], or choice sets [Sussman & Steele 80] [Waltz 75].



Given the diversity of constraints in the job shop scheduling domain, it is necessary to provide a variety of forms for specifying *relaxations* (i.e. alternative values) of constraints. Accordingly, relaxations may be defined within the ISIS constraint representation as either predicates or choice sets, which, in the latter case, are further distinguished as discrete or continuous. However, the simple specification of bounds on a variable provides no means of differentiating between the values falling within these bounds, a capability that is required by ISIS both for generating plausible alternative schedules for consideration and for effectively discriminating among alternative schedules that have been generated to resolve a given conflict. The necessary knowledge is provided by associating a *utility* with each relaxation specified in a constraint, indicative of its preference among the alternatives available. The utility of a relaxation may have more than one interpretation, which can be problematic. In the case of a due date constraint, it represents a preference for shipping on time rather than late. In the case of shifts, it represents the degree of difficulty with which another can be added. In both cases, the focus is on the difference in utility between alternative relaxations. This difference is called the *elasticity* of the relaxation. The greater the decrease in utility, the lower the elasticity. If the information were available, the utility measure would reduce to a cost function[1].

The relative influence to be exerted by a given constraint, i.e. its *importance*, is a second aspect of the constraint representation. Not all constraints are of equal importance. The due date constraint associated with high priority orders, for example, is likely to be more important than an operation preference constraint. Moreover, the relative importance of different types of constraints may vary from order to order. In one order, the due date may be important, and in another, cost may be important. Both of these forms of differentiation are expressible within the ISIS constraint representation; the former through the association of an absolute measure of importance with each constraint, and the latter by the use of *scheduling goals* which partition the constraints into importance classes and assign weights to be distributed among each partition's members. This knowledge enables ISIS to base its choices of which constraints to relax on the relative influence exerted by various constraints.

A third form of constraint knowledge explicitly represented is constraint *relevance*, which defines the conditions under which a constraint should be applied. Given that constraints are attached directly to the schemata, slots, and/or values they constrain, constraint relevance can be determined to a large degree by the proximity of constraints to the portion of the model currently under consideration. A finer level of discrimination is provided by associating a specific procedural test with each constraint. However, there are situations in which problems arise if the applicability of constraints is based solely on their context sensitivity to the current situation. First, many constraints tend to vary over time. The number of shifts, for example, fluctuates according to production levels set in the plant. Consequently, different variants of the same constraint type may be applicable during different periods of time. Within the ISIS constraint representation these situations are handled by associating a *temporal scope* with each variant, organizing the collection of variants according to the temporal relationships among them, and providing a resolution mechanism that exploits the organization. A second problem involves inconsistencies that might arise with respect to a given constraint type. Since ISIS is intended as a multiple user system, different variants of the same constraint type could quite possibly be created and attached to the same

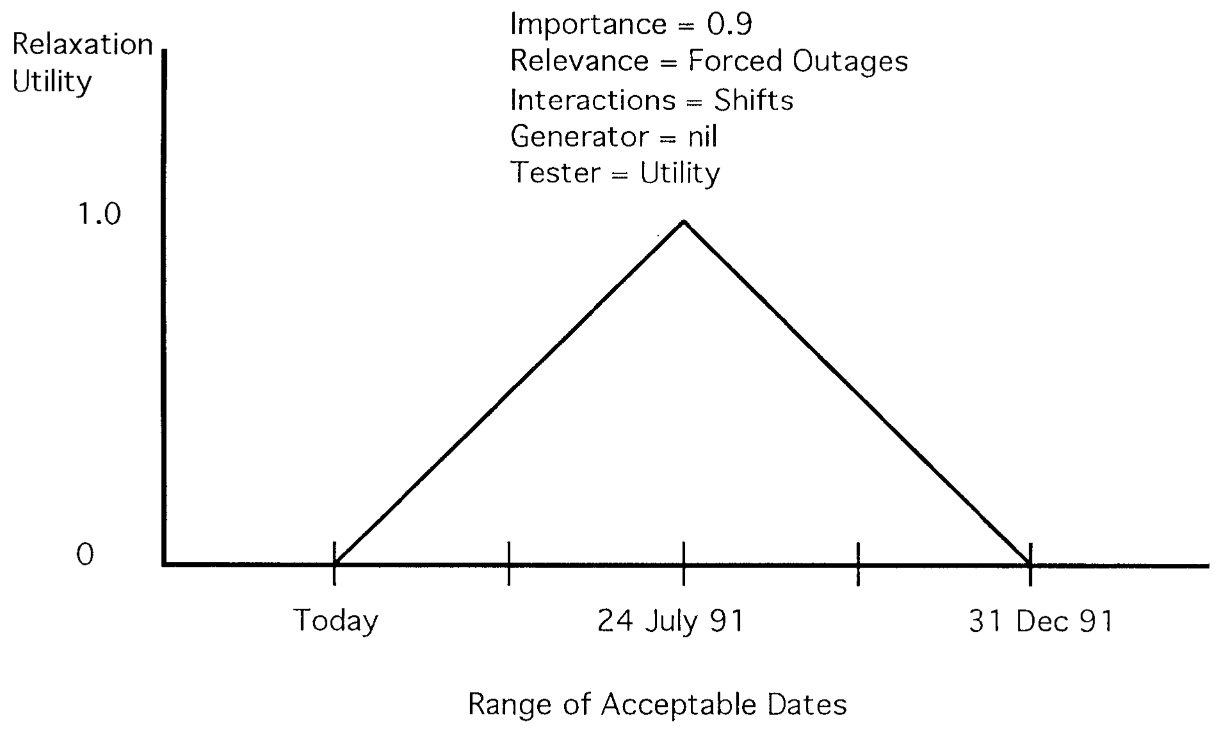
1. The use of costs as a means of ordering relaxations has been explored more recently by Sadeh [Sadeh 91].

object in the model. For example, both the material and marketing departments may place different and conflicting due date constraints on the same order. In this case, a first step has been taken to exploit an authority model of the organization in order to resolve such inconsistencies.

A fourth aspect of the constraint representation concerns the *interactions* amongst constraints. Constraints do not exist independently of one another, but rather the satisfaction of a given constraint will typically have a positive or negative effect on the ability to satisfy other constraints. For example, removing a machine's second shift may decrease costs but may also cause an order to miss its due date. These interdependencies are expressed as relations within the ISIS constraint representation, with an associated *sensitivity* measure indicating the extent and direction of the interaction. Knowledge of these interactions is used to diagnose the causes of unsatisfactory final solutions proposed by the system, and to suggest relaxations to related constraints which may yield better results.

A final concern is that of constraint *generation*. Many constraints are introduced dynamically as production of the schedule proceeds. For example, a decision to schedule a particular operation during a particular interval of time imposes bounds on the scheduling decisions that must be made for other operations in the production process. The dynamic creation and propagation of constraints is accomplished by attaching constraint generators to appropriate relations in the model.

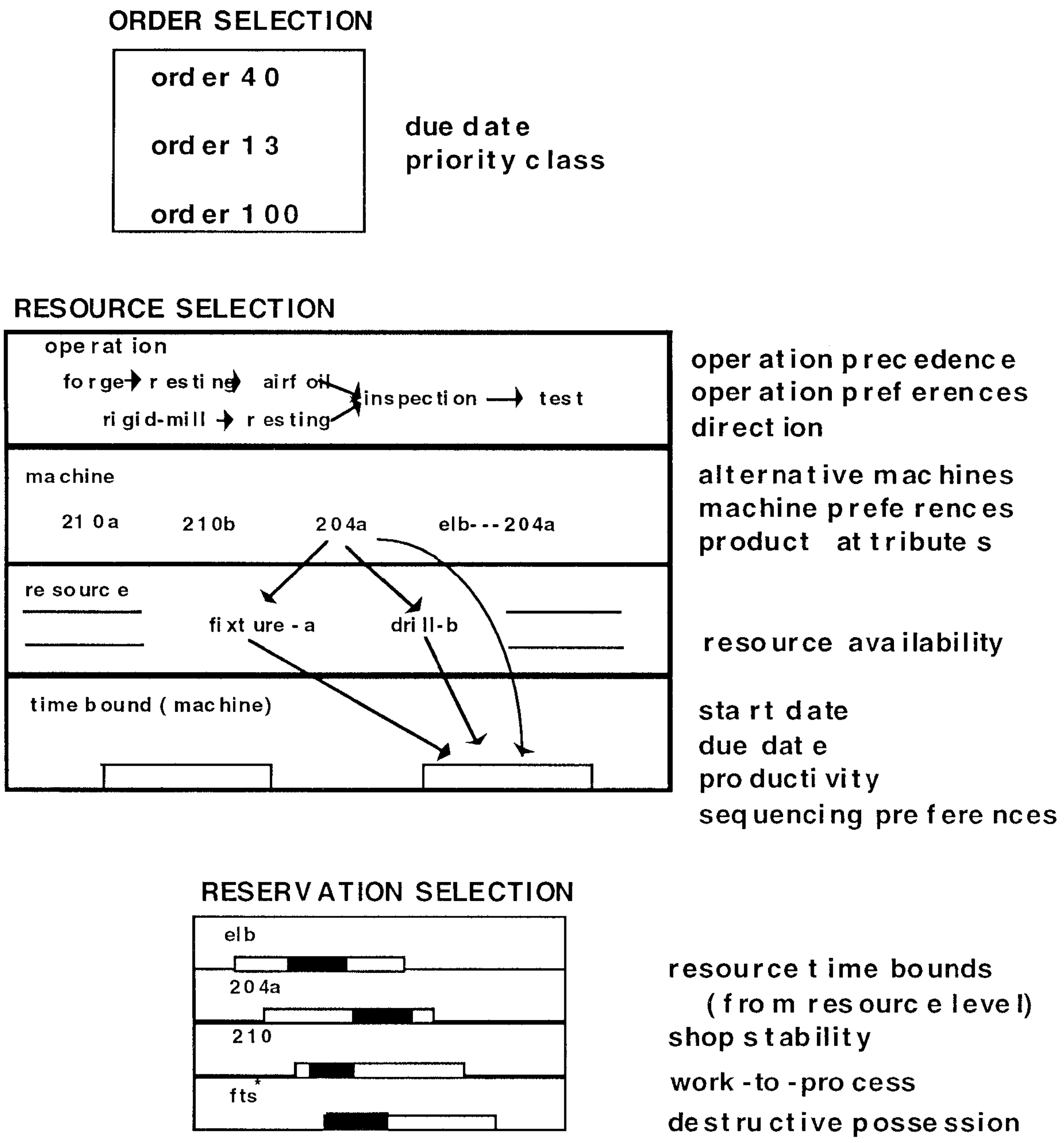
The basic due-date-constraint is a continuous value constraint which constrains the due-date slot of an order. The choice of a due-date has a utility specified by the piece-wise-linear utility function. An example of its use is a due date for forced outage orders. The tester for due-date-constraints takes the search state and the constraint as parameters, retrieves the due date being considered in the state, or predicts one if the final operation has yet to be scheduled, and applies the value of the utility function slot to the due date. The utility function uses the piece-wise-liner utility value to interpolate and return a utility.



This example specifies that the utility of the due date chosen is 1.0 if the date chosen is 24-July-91. Otherwise it has a linearly decreasing utility which bottoms out to zero if the scheduling decision is today or earlier, or after 31-Dec-92.

4.2 Constraint-Directed Scheduling

Search is divided into three levels: Order selection, resource analysis, and resource assignment. Each level is composed of three phases: A pre-search analysis phase which constructs the problem, a search phase which solves the problem, and a post-search analysis phase which determines the acceptability of the solution. In each phase, ISIS-1 uses constraints to bound, guide, and analyze the search.

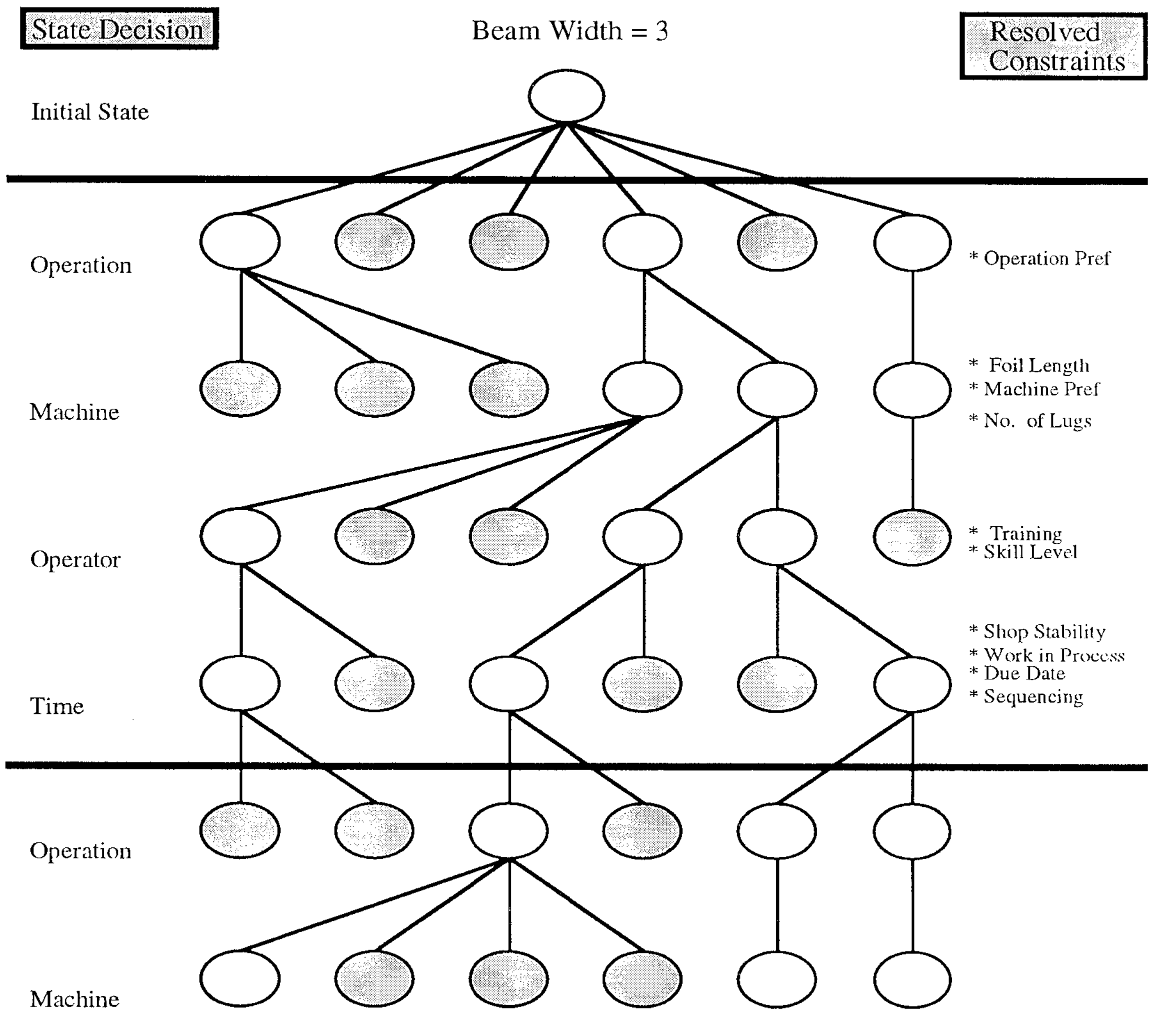


The ***order selection level*** is responsible for selecting the next unscheduled order to be added to the existing shop schedule. Its selection is made according to a prioritization algorithm that considers order type and requested due dates. The selected order is passed to the resource analysis level for scheduling.

The ***resource analysis level*** selects a particular routing for the order and assigns reservation time bounds to the resources required to produce it. Pre-search analysis begins with an examination of the order's constraints, resulting in the determination of the scheduling direction (either forward from the start date or backward from the due date), the creation of any missing constraints (e.g. due dates, work-in-process), and the selection of the set of search operators which will generate the search space. A beam search is then performed using the selected set of search operators. The search space to be explored is composed of states which represent partial schedules. The application of operators to states results in the creation of new states which further specify the partial schedules under development. Depending on the results of pre-search analysis, the search proceeds either forward or backward through the set of allowable routing for the order. An operator that generates states which represent alternative operations initiates the search, in this case generating alternative initial (or final) operations.

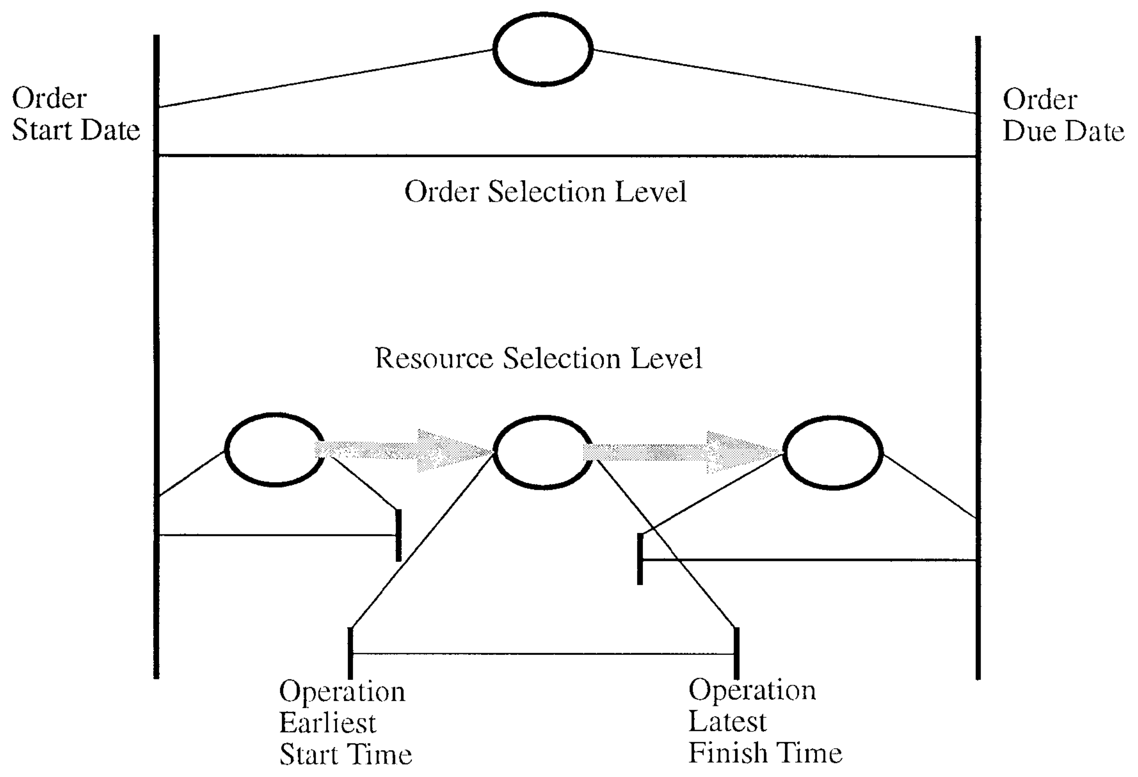
Once a state specifying an operation has been generated, other operators extend the search by creating new states which bind a machine and/or execution time to the operation. A variety of alternatives exist for each type of operator. For example, two operators were tested for choosing the execution time of an operation. The *eager reserver* operator chose the earliest possible reservation for the operation's required resources, and the *wait and see* operator tentatively reserved as much time as available, leaving the final decision to resource selection level. This enabled the adjustment of reservations in order to reduce work-in-process time. Alternative resources (e.g. tools, materials, etc.) are generated by other operators.

Each state in the search space is rated by the set of constraints found (resolved) to be relevant to the state and its ancestors. This set is determined by collecting the constraints attached to each object (e.g. machine, tool, order, etc.) specified by the state and applying resolution mechanisms. Each constraint assigns a utility to a state. The rating of a state with multiple constraints is the mean of the utilities assigned by the constituent constraints, each weighted by the the importance of the assigning constraint.



Once a set of candidate schedules has been generated, a rule-based post search analysis examines the candidates to determine if one is acceptable (a function of the ratings assigned to the schedules during the search). If no acceptable schedules are found, then diagnosis is performed. First, the schedules are examined to determine the types of scheduling errorr and the appropriate repairs. Intra-level repair may result in the re-instantiation of the level's search. For example, if the order was of high priority and was scheduled backwards from its due date, and if its start date turned out to be earlier than "today", then the order would be rescheduled at high priority in the forward direction from "today". Pre-analysis is performed again to alter the set of operators and constraints for rescheduling the order.

This level outputs reservation time bounds for each resource required for the operations in the chosen schedule. A time bound is a time interval in which an operation should be scheduled at the next level. The following figure depicts how one level constrains the temporal decisions of another. The Order Selection Level constrains the Resource Selection Level by providing a time bound for the first operation of the Order Start Date, and the Order Due Date for the last operation. The Resource Selection Level then refines the order time bound to individual time bounds for each of the operations. Note that time bounds may overlap due to resource availability.



The ***resource selection level*** establishes actual reservations for the resources required by the selected operations which minimize the work-in-process time. The algorithm takes the time bounds for each resource and proceeds to shift the availability of the resources within the bounds so that a schedule is produced which minimizes work-in-process time.

4.3 Performance

Experiments were performed with a real plant model and order data. In each experiment, an empty job shop was loaded with a representative set of 85 orders with arrival times distributed over a period of two years. The various types of constraint knowledge influencing the development of schedules in these experiments included alternative operations, alternative machines, requested due dates, requested start dates, operation time bounds, order priority classification (with orders falling into 4 priority classes), work-in-process restrictions, queue ordering constraints to reduce setup time, machine constraints on product form and length, resource availability, and shop stability (minimizing pre-emption).

A number of experiments were performed. These experiments explored the effects of alternative constraints, alternative search operators, and beam width size. A detailed discussion of all experiments may be found in [Fox 83] [Fox 87].

The following gantt chart depicts a schedule created by ISIS-1. Each row represents a machine, and each column a week. If a position in the gantt chart is empty, then the machine is idle for that week. If a position contains an "o", then it is utilized for less than 50% of of its capacity. If the position contains a "@", then over 50% of its capacity is utilized. Machines that are encountered earlier in the process routings appear closer to the top of the chart.

```
Machine
ffs* | @             @@@@O OO @ @O@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@O@@@@O@O
xao* |                                                         OO        @O
msr* |                           O@@@@O@@               O@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@O
mr3a |                OO@O O      O@  @@@@ @OOO@@     O   O OO@OO  O@@@@@@O O@O@O@@@
fss* |                  O              @OO OOO            O O OO@@ OO@@O@@O@@@@O@OO@@@@@@@O
wac1 |                                  OOOO                  O   O    OOOOO
wac2 |                                                                O@O             O
wac3 |                                                                @  O@  OOO O@ O@O@O@
wac4 |                  OO            O@O  O@@            O@ O O@OOOOO@OO O@@OO@@@@OO@@@@@@
wsf1 |                                                                                  @@
wmf2 |
wsf3 |                                                                       @@O   O@@O
wsf4 |                                                                @O   @O@@O @@@   O@@
mvt1 |                                                                 @O     @@O  @ @@OOO
sabb |                                  O     O                         O    OO   O OO  OO
fps* |                   O            OO  @ #OOO           OO    @@OOO@@@O@@@@@@@@@@@@@@O@@@@O
m3db |                                  O OOOO                     OO    @ O    OO O@ O@   O O
xpma |                                                           OOO  OOOO   OOOOOOO OOOOOO@O
gbbw |                                                                            OO O  O
jwia |                                                                            @   OO  O
jwib |                                                                        O @ O   @O OOO
jwic |                                                                O@O     @OO OOOO@@@@@OO
ftq* |                                                            @@    O@@@  O@O@@@@@O@@@@@O@@@@@@@@@@@@@@@
fts* |                                 OO OO@@@@@@@@         @@@@@@@@@ @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
flpa |                                                                    O O OO      OOOO OO OOOOO OO  OO OOO  OOO OO  O O OO   O OOO O
```

The schedule is a poor one; 65 of the 85 orders scheduled were tardy. To compound the problem, order tardiness led to high work-in-process times (an average of 305.15 days) with an overall makespan@foot[Makespan is the time taken to complete all orders.] of 857.4 days. These results stem from the inability of the beam search to anticipate the bottleneck in the "final straightening area" of the plant (the fts* machine on the gantt chart) during the early stages of its search. Had the bottleneck operation been known in advance, orders could have been started closer to the time they were received by the plant and scheduled earlier through the bottleneck operation.

Beam search sizes between 5 and 20 were tested. Sizes greater than 10 had little affect on the outcome, while sizes less than 10 performed more poorly.

4.4 Search Concepts

Constraints as Generators. Constraints which specify precedence between operations and requirements for resources can be interpreted as search operators. For example, a constraint that specifies that drilling must following milling can be interpreted as an operator that extends a state, for which milling is defined to be the operation, into a new state for which drilling is the successor operation. Some of the constraints in ISIS-1 have code that interprets the constraint as an operator to be used in search. ISIS-1's presearch analysis selects the operators from a subset of available constraints.

Constraints Bound the Search Space. The omission of constraints by pre-search analysis (e.g., not including an operator that generates alternative shifts), when selecting operators, results in a bounding of the search space. This restriction on the size of the search space is intentional but can be relaxed by post-search analysis.

Generative Constraint Relaxation. The joint satisfaction of all constraints simultaneously is impossible due to conflict among constraints. Relaxation is the process by which alternative solutions are explored by considering values for variables that are relaxations of their corresponding constraints. Generative relaxation is one type of relaxation process. Relaxations of constraints are generated during the search process by extending the code which interprets a constraint as an operator. It uses the specified relaxations to generate alternative successor states that define alternative bindings of variables. In some cases, the number of relaxations are large (e.g., a continuous constraint such as start time of an operation), requiring the code to use a relaxation's utility to determine whether it is good enough to be generated.

Constraint Resolution and Dynamic Evaluation. ISIS-1 extends the concept of dynamic evaluation function construction by utilizing a more sophisticated form of constraint resolution. *Local Resolution* is the process by which ISIS-1 dynamically resolves the set of applicable constraints at each search state. Resolution is performed by examining each schema (i.e., operation, machine, etc.) in the current state description. The contents of any constraint slots, or constraints attached to any slots that enable the object are added to the local resolution set. Constraints may originate from four sources:

- **Model-Based:** Constraints may be embedded in any resource or activity in the factory model. For example, there may be physical constraints associated with a machine, sequencing constraints associated with an operation, queue ordering constraints associated with certain work centers.
- **Lateral Imposition:** Constraints can also be propagated laterally during the search. A decision made earlier in the elaboration of a schedule may result in a constraint being attached to the lot that restricts a choice point further on in the search. For example, the selection of one operation over another early in the process plan may preclude other operations later in the process plan.
- **Exogenous Imposition:** The user may also create and implant constraints. These constraints can be attached to anywhere in the model, or be globally attached so that it is considered at each search state.

Global Resolution. The rating of a state is a rating of the partial schedule up to the current state, and not the single choice represented by the state. Hence, the rating of a state must include not only the local constraints but the constraints applied to all the states along the partial schedule ending at the current state. Not all the constraints locally resolved at each state along the path are globally resolved. Consider the due-date-constraint. It is a classic evaluation function as defined in heuristic search. Part of the constraint calculates the work-in-process time of the lot to the current state, and the other part predicts the remaining work-in-process time to the end state. Each time the constraint is applied, it is a better estimator of the work-in-process time, and should override applications of the same constraint earlier in the partial schedule. On the other hand, the queue-stability constraint is applied at each state that reserves a machine. It rates the state by how much it destabilizes existing machine reservations. The greater the destabilization, the lower the rating. This constraint measures a decision made at that state, and remains invariant over future states, since any future states will not affect an earlier state.

Constraints are classified into two categories: *invariant* and *transient*. All invariant constraints participate in the globally resolved constraint set, and only the most recent version of transient constraints participate.

Relative Resolution. All constraints are not created equal. Relative resolution differentially interprets the resolved constraints by partitioning the constraint set according to the applicable scheduling goal. A scheduling goal partitions the constraint set and defines an importance for each partition. The importance is then uniformly divided amongst the constraints in the partition.

Analytic Relaxation via Constraint Diagnosis and Repair. The completion of the beam search may result in schedules which are not acceptable due to the poor satisfaction of many of its important constraints. Analytic relaxation is defined to be the process by which the results of the search are examined to determine which "peep hole" repair of a constraint will generate a significant increase it the overall constraint rating of a schedule. In addition to the procedural embedding of situational knowledge in the form of rules (e.g., IF you cannot meet the due date THEN relax the start date constraint by starting earlier), a declarative approach was taken. Each constraint may have a "constrains" relation that links it to another constraint. If the first constraint was not acceptably satisfied (e.g., due date), then by searching along the "constrains" relation another constraint could be found (e.g., shifts) whose further relaxation or strengthening could impact the first constraint. Consequently, post-analysis could suggest the increase in number of shifts to pre-search analysis and have the search re-run.

5.0 ISIS-2: Hierarchical Constraint-Directed Scheduling

ISIS-1 identified the representational requirements of constraints and their use in directing search. Neither changes in beam width, nor alterations to existing constraints were able to significantly affect the degree to which due date and work in process constraints were optimised. Simply stated, ISIS-1 schedules were bad. The cause of this problem lay with the combinatorics of the search space combined with the horizon effect. ISIS-2 was designed to reduce the impact the horizon effect had on the quality of the schedules [Fox 83] [Fox 87]. ISIS-2 constructs schedules by performing a hierarchical, constraint-directed search in the space of alternative schedules.

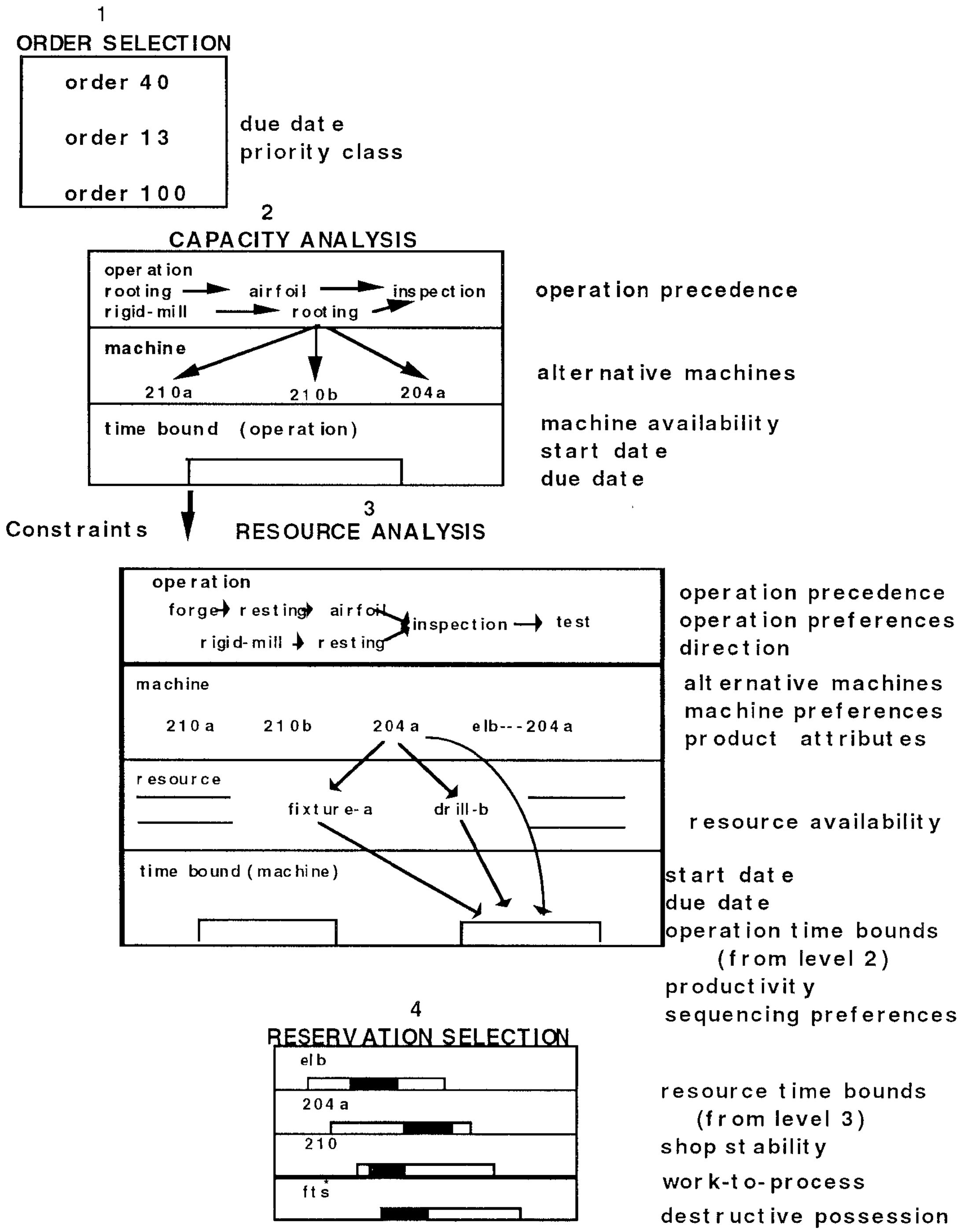
5.1 Architecture

An analysis of the schedules produced by ISIS-1 led to the conclusion that orders were spending too much time waiting for bottlenecked resources, i.e., machines. Given that demand for capacity exceeded availability, if an order was to be completed on time, it should either avoid the bottlenecks by choosing an alternative path in their process routing, or reach the bottleneck earlier. The question was how to do this.

Given the anchoring of ISIS-1's search, (i.e., it searched forward from the first operation or backwards from the last operation, but not opportunistically), how could we provide information that would allow the resource selection level search evaluation function to ascertain the probable impact of its decision? If we could determine the latest start time for each operation in an order's process plan that would allow it to complete by its due date, then we could provide this to the beam search in the form of a constraint. Then any state whose start time was later than specified by the "time bound" constraint would receive a low rating.

The calculation of these time bounds could be done efficiently by ignoring constraints and resources that had little impact on the bottleneck problem. That is, the computation could be done on an *abstraction* of the original problem. The latest start time of an order's final operation would be determined by an analysis of the available capacity of all the machines an operation may use prior to the order's due date. Once the latest start time of the final operation is determined, earlier operations' latest start times could be determined in a similar manner assuming their subsequent operation provides them with a finish time (i.e., using dynamic programming). To achieve this, an additional level was added between order selection and resource selection: *capacity analysis*. This level considers a subset of the more important constraints in order to "look ahead" so that capacity bottlenecks could be identified in a smaller search space.

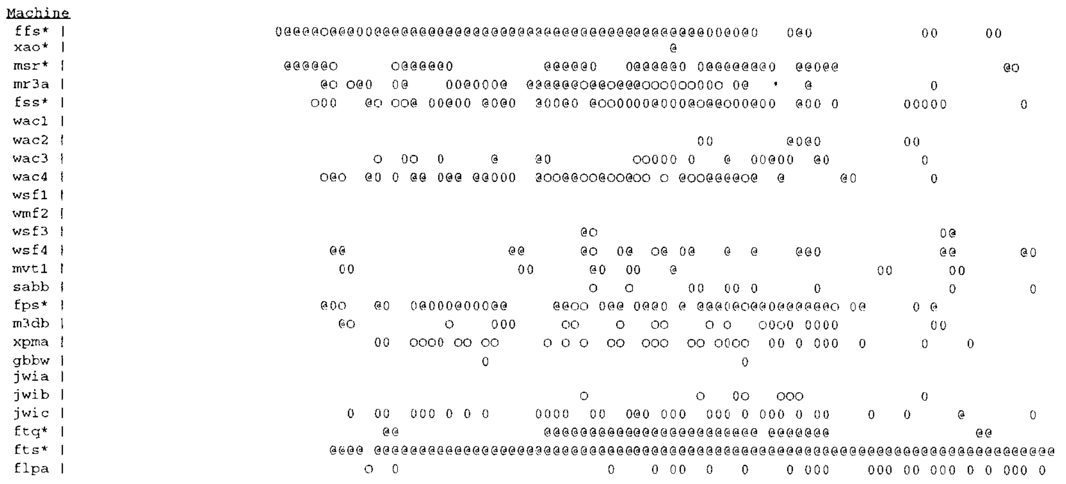
Capacity analysis takes as input the selected order from the order selection level and uses the following subset of constraints in its search: due date, start date, operation precedence and alternatives, machine requirements, and machine reservations. All other constraints are ignored. The capacity analysis level performs a dynamic programming analysis, i.e., critical path analysis, of the plant based on current resource capacity constraints and existing resource reservations. It determines the earliest start time and latest finish time for each operation of the selected order, as bounded by the order's start and due date. The times generated at this level are codified as operation time bound constraints which hierarchically propagated to the resource analysis level.



Post analysis of search was extended to include "inter-level repair." It is initiated when diagnosis determines that the poor solutions were caused by constraint satisfaction decisions made at another level. Inter-level diagnosis can be performed by analyzing the interaction relations linking constraints. A poor constraint decision at a higher level can be determined by the utilities of constraints affected by it at a lower level, and an alternative value may be chosen.

5.2 Performance

ISIS-2's inclusion of a level of abstraction in the top down search hierarchy had a significant impact on the results, evidenced by the increased satisfaction of the due date constraints.



The average utility assigned by the due date constraint to lower priority "service orders", for example, almost doubled, rising from a value of 0.46 in the first experiment to a value of 0.80. The total number of tardy orders was reduced to 14. Moreover, a much lower average work-in-process time of 186.73 days was achieved, resulting in an overall makespan of 583.25 days. In this case, inadequate machine capacity in the "final straightening area" (fts*) appeared to be the principal limitation affecting order tardiness.

5.3 Search Concepts

Periscoping. The improved performance of ISIS-2 rests on the ability of the capacity analysis level to identify bottlenecks and encode their effect in the form of operation time bound constraints. At the resource selection level, whenever alternatives are generated for the time to perform a particular operation, the operation time bound constraint is resolved and evaluated. The effect is what we call *periscoping*. It is as if the evaluation of the state looked "up above" the local situation to see what problems lay further down the search path it has yet to explore. If there was a bottleneck, the operation time bound constraint would lower the utility of times which do not provide enough time to get through the bottleneck.

Constraint-Directed Focus of Attention. The hierarchical imposition of constraints of one level onto the next results in the lower level's focusing of its search on the "better" parts of the search space, reducing the complexity of the search while increasing the utility of the outcome.

Constraint Stratification. Constraints appear to fall naturally into a partial ordering in this domain according to the degree of difficulty with which they can be relaxed. For example, it is easier to alter a due date by a day than it is to add another shift. Consequently, a level of the search hierarchy, in addition to constraining the search of the next level via periscoping, can determine the values of a subset of constraints which are more difficult to change than constraints at a lower level. This concept will be elaborated in the next section.

Interlevel Analytic Relaxation. The concept of analytic relaxation is extended to work across levels. If a constraint is identified as needing to be relaxed, and it is bound at a higher level, then post-search analysis could re-invoke the higher level. The level will either alter the constraint and/or perform the search again at that level.

5.4 Reactive Scheduling in ISIS-2

From early in the project, the issue of being able to respond to dynamic changes on the factory floor was recognized as an important part of a scheduling system. Machine breakdowns, material shortages, unavailable tooling and personnel illness all conspired to make any scheduling generated by ISIS invalid within hours. Though much of our effort was focused elsewhere, we did implement a schedule repair method.

1. Whatever the source of change, the final impact was to invalidate one or more activity reservations.
2. ISIS then identified the activities affected by the change.
3. For each activity, ISIS identified the order it was associated with and retracted the reservation and the remaining downstream reservations for that order.
4. ISIS then turned each of the retracted reservations into a preference constraint.
5. It then rescheduled each affected order.

With this approach, only the affected reservations and those subsequent to them were rescheduled. By turning the retracted reservations into preferences, the scheduler attempted to construct a schedule as close to the initial schedule as possible. We did not run any experiments to ascertain how well this method performed.

6.0 Conclusion

The ISIS constraint-directed scheduling system was designed to provide intelligent support in the domain of job shop scheduling. Job-shop scheduling is a "uncooperative" multi-agent (i.e., each job is to be "optimized" separately) planning problem in which activities must be selected, sequenced, and assigned resources and times of execution. Resource contention is high, causing scheduling decisions to be closely coupled. Search is combinatorially explosive; for example, 85 orders moving through 10 operations without alternatives, with a single machine substitution for each and no machine idle time has over 10^{1000} possible schedules. The selection of a schedule is influenced by such diverse factors as due date requirements, cost restrictions, production levels, machine capabilities and substitutability, alternative production processes, order characteristics, resource requirements, and resource availability.

At the core of the ISIS family of systems is an approach to automatic scheduling that provides a framework for incorporating the full range of real world constraints. Given the conflicting nature of the domain's constraints, the problem differs from typical constraint satisfaction problems. One cannot rely solely on propagation techniques to arrive at an acceptable solution, since no feasible solution may exist. Instead, constraints must be selectively *relaxed*; the problem solving strategy becomes one of finding a solution that best satisfies the constraints. Secondly, scheduling is an optimisation problem where we seek to find the best schedule. Thus, the design of ISIS has focused on

- Constructing a knowledge representation that captures the requisite knowledge of the job shop environment and its constraints to support constraint-directed search, and
- Developing a search architecture capable of exploiting this constraint knowledge to effectively control the combinatorics of the underlying search space, while at the same time finding the best solution possible.

This results in a system that can generate detailed production schedules that accurately reflect the current status of the shop floor, distinguishing ISIS from traditional scheduling systems that are more myopic.

The perspective that ISIS introduced, that scheduling is a constraint-directed search process, has been embraced within AI and outside of it. A recent paper by McKay [McKay 93] documents the plethora of constraints in the "factory from hell" from an Operations Research perspective. Even Goldratt, the creator of OPT, publishes a journal on "constraints". The methods of reasoning about constraints have grown more sophisticated since ISIS was completed [Ow & Smith 87] [Fox et al. 89] [Keng & Yun 89] [Sadeh 92] [Zweben et al. 90], however, ISIS's ability to represent and use all of a domain's constraints still endures.

7.0 Acknowledgements

The work reported in this paper represents the effort of many of my colleagues who were at CMU at the time: Steve Smith, Brad Allen, Ranjan Chak, Lin Chase, Steve Miller, Tom Morton, Peng Si Ow, Ram Rachamadugu, Gary Strohm, Ari Vepsalainen, Mark Wright, Doug Zimmerman; and our colleagues at Westinghouse: Bob Baugh, Jose Isasi, and Dwight Mize.

This research was supported by numerous agencies and companies including the Air Force Office of Scientific Research under contract F49620-82-K0017, the Defense Advance Projects Agency under contract #F30602-88-C-0001, Westinghouse Electric Corporation, International Business Machines, Schlumberger Co., Boeing Computer Services Co., and McDonnell Douglas Corporation.

8.0 References

[Brown 86] Brown, R.
A solution to the Mission Planning Problem.
In Proceedings of the Second Aerospace Applications of Artificial Intelligence. 1986.

[Chapman 87] Chapman, D.
Planning for Conjunctive Goals.
Artificial Intelligence 32:333-377, 1987.

[Fox 83] Fox, M.S.
Constraint-Directed Search: A Case Study of Job-Shop Scheduling.
PhD thesis, Carnegie Mellon University, 1983.
CMU-RI-TR-85-7, Intelligent Systems Laboratory, The Robotics Institute, Pittsburgh,PA.

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

[Fox et al. 89]
Fox, M.S., Sadeh, N., and Baykan, C.
Constrained Heuristic Search.
In Proceedings of the International Joint Conference on Artificial Intelligence, pages 309-316. Morgan Kaufmann Pub. Inc., 1989.

[Garey & Johnson 79]
Garey, M.R., and Johnson, D.S.
Computers and Intractability: A Guide to the Theory of NP-Completeness.
Freeman and Co., 1979.

[Keng & Yun 89]
Keng, N.P., and Yun, D.Y.Y.
A Planning/Scheduling Methodology for the Constrained Resource Problem.
In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 998-1003. 1989.

[Lowerre & Reddy 90]
Lowerre, B.T., and Reddy, R.
The HARPY Speech Understanding Systems.
Trends in Speech Recognition.
In Lea, W.A.,
Prentice-Hall, Englewood Cliffs, N.J., 1990.

[McKay 93] McKay, K.N.
The Factory From Hell - A Modelling Benchmark.
In *Proceedings of the NSF Workshop on Intelligent, Dynamic Scheduling*, pages 97-113. 1993.

[Ow & Smith 87]
Ow, P-S., and Smith, S.F.
Two Design Principles for Knowledge-based Systems.
Decision Sciences 18(3):430-447, 1987.

[Sadeh 91] Sadeh, N.
Look-ahead Techniques for Micro-opportunistic Job Shop Scheduling.
PhD thesis, School of Computer Science, 1991.

[Simon 72] Simon, H.A.
On Reasoning About Actions.
Representation and Meaning: Experiments with Information Processing Systems.
Prentice-Hall, 1972, pages 414-430.

[Stefik 81] Stefik, M.
Planning with Constraints (MOLGEN: Part 1).
Artificial Intelligence 16(2):111-140, 1981.

[Sussman & Steele 80]
Sussman, G.J. and Steele Jr., G.L.
CONSTRAINTS: A language for expressing almost-hierarchical descriptions.
Artificial Intelligence 14(1):1-39, 1980.

[Waltz 75] Waltz, D.
Understanding line drawings of scenes with shadows.
The Psychology of Computer Vision.
In Winston P.H.,
McGraw-Hill New York, 1975.

[Zweben et al. 90]
Zweben, M., Deale, M., and Gargan, R.
Anytime.Rescheduling.
In *Proceedings of the Annual Conference of the Association for Artificial Intelligence.* 1990.