ISIS — a knowledge-based system for factory scheduling

Mark S. Fox and Stephen F. Smith
The Robotics Institute, Carnegie-Mellon University,
Pittsburgh, Pennsylvania 15213, USA

Abstract: Analysis of the job shop scheduling domain has indicated that the crux of the scheduling problem is the determination and satisfaction of a large variety of constraints. Schedules are influenced by such diverse and conflicting factors as due date requirements, cost restrictions, production levels, machine capabilities and substitutability, alternative production processes, order characteristics, resource requirements, and resource availability. This paper describes ISIS, a scheduling system capable of incorporating all relevant constraints in the construction of job shop schedules. We examine both the representation of constraints within ISIS, and the manner in which these constraints are used in conducting a constraint-directed search for an acceptable schedule. The important issues relating to the relaxation of constraints are addressed. Finally, the interactive scheduling facilities provided by ISIS are considered.

1. Introduction

The construction of schedules to govern the production of orders in a job shop is a complex problem that is influenced by knowledge accumulated from many different sources in the shop. The acceptability of a particular schedule depends on such diverse and conflicting factors as due date requirements, cost restrictions, production levels, machine capabilities and substitutability, alternative production processes, order characteristics, resource requirements, and resource availability. The problem is a prime candidate for application of artificial intelligence (AI) technology, as human schedulers are overburdened by its complexity and existing computer-based approaches to automatic scheduling incorporate only a fraction of the relevant scheduling knowledge. They use a purely *predictive* approach to scheduling; based on a restricted model of the environment, predictions are made as to when operations are to be performed. The resulting schedules often bear little resemblance to the actual state of the factory, leaving detailed scheduling to the shop-floor supervisor. Automatic scheduling is thus reduced to weekly or monthly runs whose outputs provide guidance in determining future loadings. By viewing the scheduling problem from an AI perspective, it is possible to provide a better solution. Schedule construction can be cast as a constraint-directed activity that is influenced by *all* relevant scheduling knowledge. The

This research was supported, in part, by the Air Force Office of Scientific Research under contract F49620-82-K0017, Westinghouse Electric Corporation, and the Robotics Institute.

result is a scheduling system that possesses an additional *reactive* capability; accurate and timely schedules can be constructed in response to the actual current state of the factor. This latter approach is the basis of ISIS, a knowledge-based decision support system for job shop scheduling [1, 2, 3].

Formulating the job shop scheduling problem as a constraint-directed activity raises some interesting research issues. Given the conflicting nature of the domain's constraints, the problem differs from typical constraint satisfaction problems and one cannot rely solely on propagation techniques (e.g. [4, 5, 6, 7]) to arrive at an acceptable solution. Rather, constraints must be selectively *relaxed* and the problem-solving strategy must be one of finding a solution that best satisfies the constraints. This implies that constraints must serve to discriminate among alternative hypotheses as well as to restrict the number of hypotheses generated. 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.

The remainder of this paper focuses on the issues surrounding the constraint-directed reasoning approach to job shop schedule construction taken in the design of ISIS. In Section 2 we examine the nature and complexity of the job shop scheduling problem within an actual manufacturing facility. The wide variety of constraints that influence job shop schedules are identified and categorized. This is followed in Section 3 by a description of the constraint representation used to characterize this knowledge within ISIS. In Section 4, the issues surrounding the use of constraint knowledge are addressed. The constraint-directed search conducted by ISIS to automatically construct job shop schedules is discussed, and some performance results are presented and discussed in Section 5. We turn our attention to the practical capabilities provided by the ISIS interactive scheduling subsystem in Section 6. The additional features of ISIS are summarized in Section 7. Finally, in Section 8, we draw some conclusions based on our experience with the system.

2. The scheduling problem

The job shop scheduling problem can be defined as selecting a sequence of operations (i.e., a process routing) whose execution results in the completion of an order, and assigning times (i.e., start and end times) and resources to each operation. Historically, the scheduling problem has been divided into two separate steps. Process routing selection is typically the product of a planning process, while the assignment of times and resources is typically the purpose of scheduling. Actually, the distinction between planning and scheduling is somewhat fuzzier, as the selection of a routing cannot be made conclusively without generating the accompanying schedule. The admissibility of a process routing depends on the feasibility of each selected operation, and a given operation is feasible only if its resource requirements are satisified during the time that the operation is to be performed. Thus, the determination of an admissible process routing implies a prior assignment of resources and times to each operation in the routing.

The job shop scheduling problem has been described as NP-hard. Consider the sequencing of just ten orders through five operations. Associating a single machine with each operation (i.e. no alternative routings) and assuming no time gaps in the schedule to be generated, there are $(10!)^5$ or about 10^{32} possible schedules. The situation within an actual manufacturing facility is much more complex. The number of orders, operations, and resources are substantially greater, and the dynamic nature of the shop (e.g. machine breakdowns, order changes, etc.) further complicates the selection process. To illustrate the full complexity of the problem, let us examine a specific job shop scheduling environment.

2.1 The WTCP job shop environment

The Westinghouse Turbine Component Plant (WTCP) in Winston-Salem N.C. was selected as a test domain for investigating the job shop scheduling problem. The primary product of the plant is steam turbine blades. A turbine blade is a complex three-dimensional object produced by a sequence of forging, milling and grinding operations to tolerances of a thousandth of an inch. Thousands of different blade styles are produced in the plant, many of them to be used as replacements in turbines currently in service.

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

- Forced outages (FO): Orders to replace blades which malfunctioned during operation. It is important to ship these orders as soon as possible.
- Critical replacement (CR) and Ship Direct (SD): Orders to replace blades during scheduled maintenance. Advance warning is provided, but the blades must arrive on time.
- Service and shop orders (SO,SH): Orders for new turbines. Lead times of up to three years may occur.
- Stock orders (ST): Orders for blades to be placed in stock for future needs.

The area of the plant considered by ISIS has 100 to 200 orders in process at any time.

Turbine blades are produced according to a process routing or lineup. A routing specifies a sequence of operations that leads to the finished product. An operation is an activity which defines the resources required (e.g. machines, tools, materials and fixtures), machine set up and run times, and labor requirements. Each type of turbine blade produced in the plant has one or more process routings, each containing ten or more operations. Distinctions between process routings may be as simple as substituting a different machine, or as complex as changing the manufacturing process. The resources needed for an operation are typically shared with other operations in the shop.

During our discussions with WTCP, we found that orders are not scheduled in a uniform manner. Each scheduling decision to be made entails side effects whose importance varies by order, and the generation of a schedule is an iterative process. What quickly became evident was the scheduler's reliance on information other than due dates, process routings, and machine availability. A proposed schedule is distributed to persons in every department in the plant, and each person on the distribution list can provide information which may result in schedule alterations. We found that the scheduler spends only 10%–20% of his time actually scheduling, and 80%–90% of his time communicating with other employees to determine what additional 'constraints' should influence an order's schedule.

From this analysis, we may conclude that the objective of scheduling is not only meeting due dates, but satisfying the many constraints that originate from various parts of the plant. Scheduling is not a distinct function, separate from activities in the rest of the plant, but is highly dependent upon decisions being made elsewhere in the plant. The added complexity imposed by these constraints leads schedulers to produce schedules that are characterized by high work-in-process times, order tardiness, and low machine use. What is needed is a general methodology for using such diverse sources of information in the generation of job shop schedules.

2.2 Constraint categories

Any attempt to provide a general solution to the job shop scheduling problem must begin with an identification of both the set of scheduling constraints to be considered and their effect on the scheduling process. Our analysis of the constraints present in the WTCP plant has yielded five broad categories of constraints. These categories are examined below.

The first category of constraint encountered in the factory is what we call 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-progress*: Work-in-progress (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 workforce.
- *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 is, 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 that it can be used for. 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 predence 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 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 *prefer-*

Organizational goals	Due date Work-in-process Shop stability Shifts Cost Productivity goals Quality
Physical constraints	Machine physical constraints Set-up times Processing time Quality
Causal restrictions	Operation alternatives Machine alternatives Tool requirements Material requirements Personnel requirements Inter-operation transfer times
Availability constraints	Resource reservations Machine down time Shifts
Preference constraints	Operation preferences Machine preferences Sequencing preferences

Figure 2.1: Scheduling constraints

ence. 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.

Figure 2-1 lists the variety of constraints we have identified as well as the categories we have used to classify them.

In a review of commercial scheduling systems we found that most of them provide only simple capacity analysis with an emphasis on meeting due dates. Little or no consideration is given to providing general facilities for representing and using any additional constraints. Moreover, these systems are batch oriented, and meant to be run weekly or monthly; they do not provide real-time control. This was found to be unacceptable by WTCP. On the other hand, management science research has focused on optimal results for artificial problems, or dispatch rules for meeting due dates or maximizing facility use. These solutions are also found to be unsatisfactory for the real-life scheduling problem.

3. Modeling the domain and its constraints

The wide variety of constraints identified above indicate the need for a rich underlying model of the job shop scheduling domain. Detailed knowledge of all facets of the domain, including operations, process routings, machines, work areas, tools, materials, personnel, orders, etc. must be accessible to the scheduling system if it is to intelligently construct job shop schedules. The characterization of this knowledge within ISIS is considered in the following subsections. The ISIS modeling system and its constraint representation are described in turn.

3.1. The ISIS modeling system

The ISIS modeling system contains all the knowledge necessary to plan and schedule production in a job shop environment. The system is built using SRL [8, 9], a flexible knowledge representation system which allows the user to mould the language to his needs. SRL is a frame-based language which encodes concepts as schemata. A schema is a collection of slots and values. Each schema, slot, and/or value may have meta-information attached to it. In addition to attribute knowledge, slots define inter-schema relations, along which slots and values may be inherited. The inheritance semantics of

```
{{ operation
    { IS-A act
        NEXT-OPERATION: "operations which follow this"
        PREVIOUS-OPERATION: "operations which directly precede this"
        ENABLED-BY: "state which enables this action"
        CAUSES: "states caused by this action"
        DURATION: "time of this action"   }    }}
```

Figure 3.1: Operation schema

a relation are user-definable. Figure 3.1 illustrates the basic SRL construct in defining an operation schema.

In this case, the description states that an *operation* IS-A type of act. This view of an operation is further refined to include the attributes (slots) NEXT-OPERATION, PREVIOUS-OPERATION, etc. SRL has been used to support a number of different Intelligent Management System functions [10] including simulation, diagnosis, graphics, project management, and long range planning.

SRL provides ISIS with the capability of modeling a plant at all levels of detail; from physical machine descriptions, to process descriptions, to organizational structures and relations. Some of the uses made of SRL within the ISIS modeling system include:

- *Order definition* — any type of order, e.g. live, forecast, customer, manufacturing, etc., can be created, and updated interactively. New types of orders can be created as needed.
- *Lot definition* — orders may be grouped into lots, which may then be run as a unit through the plant.
- *Resource definition* — resources such as machines, tools, fixtures, materials, and personnel can be defined and used by an extensible set of functions. Resource definitions include substitutability in operations, and their current operation assignments in the plant.
- *Lineup (operation) definition* — the process(es) by which a product is produced may be described as an operations graph. The operations graph describes all alternative operations, processing information, and resource requirements. Operations can be described hierarchically, enabling the description of operations at varying levels of detail.
- *Work area definition* — cost centers, work areas, and any other plant floor organizations may be defined and resources, sub work areas, etc. can be assigned to them. Possible uses besides scheduling include accounting, personnel, and other functions.
- *Department definition* — departments, personnel, and any other organization structures may be defined, and linked with other parts of the model.
- *Reservation definition* — any resource may be reserved for an activity (operation). ISIS provides full reservation creation and alteration facilities with respect to both interactive and automatic scheduling.
- *Plant organization* — the plant may be described hierarchically both from an organization structure perspective, and a physical layout perspective. This is used to support functions such as color graphics displays of the plant layout.

These descriptive capabilities are provided within the ISIS modeling system through the definition of a variety of primitives for modeling manufacturing organizations. The lowest level provides the basic concepts of states, objects, and acts, related to one another by a set of temporal and causal relations. These primitives provide a general foundation upon which more domain-specific concepts such as those identified above are defined and related. In specifying a process routing, for example, manufacturing operations are defined as acts, and relations such as NEXT-OPERATION are composed from basic temporal and causal relations. Resources required by the operations are expressed as objects, with their allocation represented as states of possession by particular operations. Constraints are represented as meta-information, and can be attached to any schema, slot, or value in the model.

Figure 3.2 gives a flavor of the ISIS job shop scheduling model, schematically depicting a portion of a process routing representing the two sequential oper-

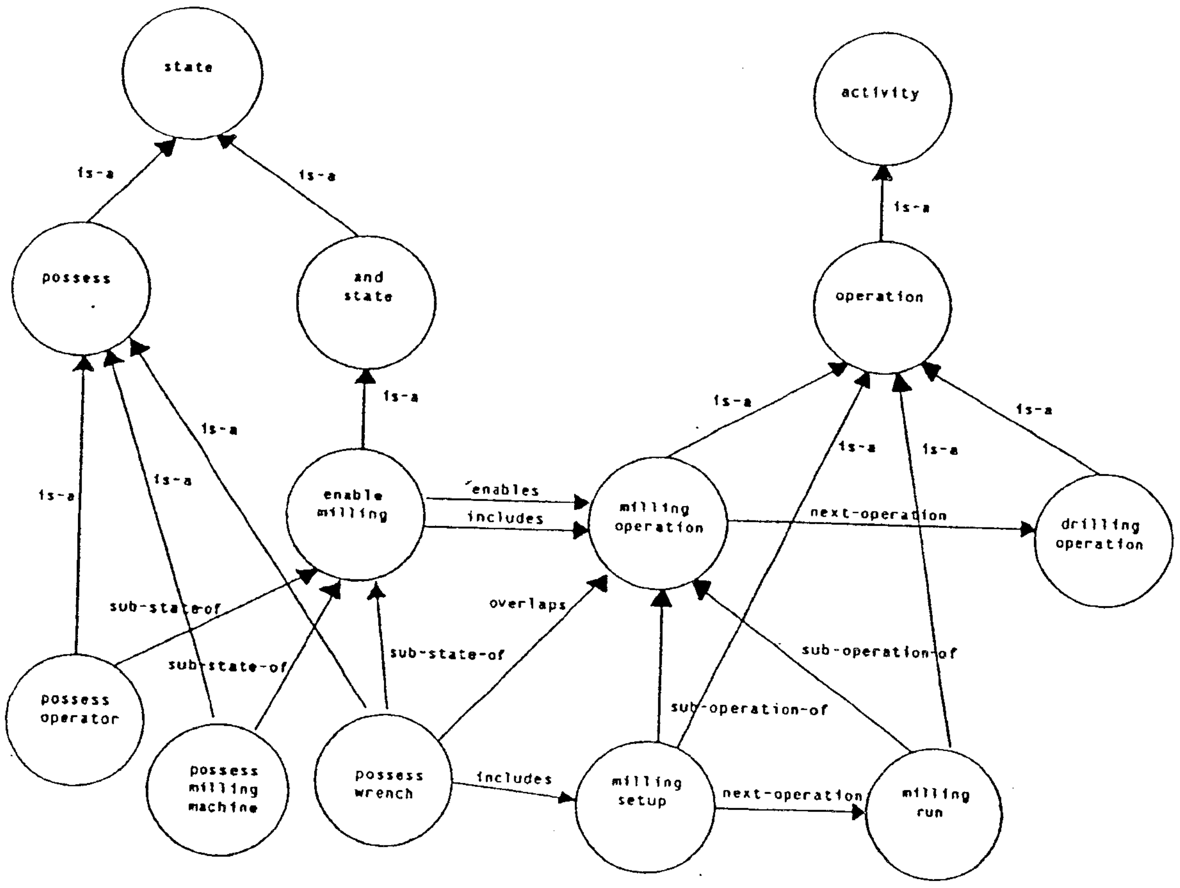

Figure 3.2: Activity model

ations of *milling* and *drilling*. The *milling* operation is defined as the composition of the sub-operations *milling-setup* and *milling-run*, with the NEXT-OPERATION relation specifying precendece between the two. Likewise, NEXT-OPERATION is used to designate *drilling* as the operation immediately following *milling*. The *milling* operation is further defined as being enabled by the *enable-milling* state. This indicates that the *enable-milling* state must exist before the *milling-operation* may be performed. The *enable-milling* state is the conjunction of sub states *possess-operator, possess-milling-machine* and *possess-wrench*, each of which is also linked to the *milling* operation via causal (e.g. ENABLES) and temporal (e.g. OVERLAPS, INCLUDES) relations.

Within the schema representation of such a model, relations appear as slots in the schemata that they relate. The *milling-operation* schema, shown in Figure 3.3, illustrates this point.

Additional schemata are present in the representation to describe the properties of the relations themselves. In the activity model of Figure 3.2, for example, the domain specific relations NEXT-OPERATION, SUB-OPERATION-OF, and SUB-STATE-OF are defined in terms of more primitive domain independent relations, forming the relation hierarchy shown in Figure 3.4

```
{{ milling-operation
    { IS-A operation
        WORK-CENTER: milling-center
        DURATION: {{ INSTANCE time-interval
                        DURATION: 5 }}
        NEXT-OPERATION: drilling-operation
        SUB-OPERATION: milling-setup milling-run
        ENABLED-BY: enable-milling }      }}
```

Figure 3.3: Milling-operation schema

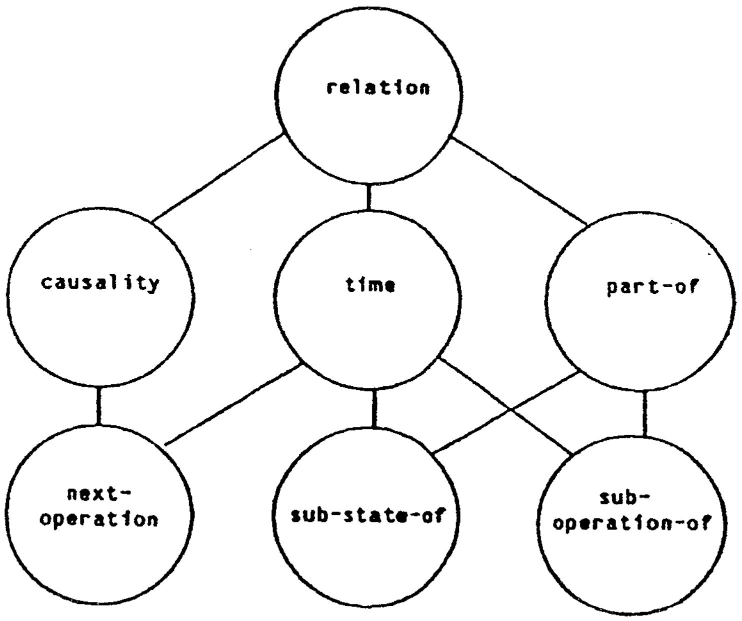

Figure 3.4: Relation hierarchy

3.2. Constraint representation

Given the role of constraints in determining a job shop schedule, a major thrust of our research has centered on identifying and characterizing 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, the implication being 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:

- what alternative dates are satisfactory if the original cannot be met?
- what preferences exist for these alternative dates?
- who specified the due date? when? and why?
- how is the satisfaction of the due date related to other constraints such as costs?
- does the satisfaction of the due date constraint positively or negatively affect the satisfaction of other constraints?
- under what circumstances should the due date constraint be considered?
- if there are two or more due date constraints specified for an order, which should be used?

Let us examine the representational issues raised by these examples, and, correspondingly, the salient features of the ISIS constraint representation (additional details may be found in [1, 2, 11]).

One of the central issues that must be addressed by the constraint representation is that of *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 may require using faster, more expensive machines, 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 usually specified by predicates [5, 12] or choice sets [6, 7].

Given the diversity in the types of constraints present 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 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 amongst 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, termed the *context*, 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 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 in exploiting an authority model of the organization 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.

Figure 3.5 summarizes the knowledge about constraints that is captured within the ISIS constraint representation.

An example of a constraint within the ISIS model is a *due-date-constraint* [Figure 3.6]. It constrains the range (i.e. value) that a slot may have. In particular, it constrains the DUE-DATE slot (relation) associated with an order schema. The specific set of alternative values (or relaxations) designated by this constraint is described by the *due-date-relaxation-spec* schema [Figure 3.7], which is defined as a type of *continuous-relaxation-spec*. A continuous choice relaxation spec restricts the value of a slot to a particular domain, in this case the domain of dates, and specifies a piece-wise linear utility function over this domain. This function provides the basis for determining the

relaxation	specification of allowable alternatives and the preferences amongst them.
importance	the relative influence to be exerted by the constraint.
relevance	the conditions under which the constraint should apply.
interaction	the constraint's interdependencies with other constraints.
generation	a mechanism for creating and propagating constraints.

Figure 3.5: Aspects of constraint knowledge

```
{{ due-date-constraint
   { IS-A range-constraint
       IMPORTANCE:
       CONTEXT: t
       DOMAIN:
           range: (type IS-A order)
       RELATION: due-date
       CONSTRAINED-BY:
           range: (type IS-A due-date-relaxation-spec) }
   PRIORITY-CLASS:   }}
```

Figure 3.6: due-date-constraint schema

```
{{ due-date-relaxation-spec
   { IS-A continuous-relaxation-spec
       DOMAIN: dates
       PIECE-WISE-LINEAR-UTILITY:
       EVALUATOR: interpolate }  }}
```

Figure 3.7: due-date-relaxation-spec schema

utility of any particular value chosen (via interpolation).

4. Constraint-directed search

ISIS has been designed to perform constraint-directed scheduling of job shops, and, as such, provides a framework for incorporating the wide range of constraints that typically influence scheduling decisions. It is able to identify which constraints to satisfy, on an order-by-order basis, and may selectively relax any constraints found to be unsatisfactory. Among the principle characteristics of the automatic scheduling system are:

- *Constraint utilisation* — the general framework enables ISIS to use almost any constraint the user deems appropriate.
- *Multi-level scheduling* — ISIS allows scheduling to be performed at differing levels of detail and perspectives. Included in the current version is a bottleneck analysis whose output is a set of constraints to guide the detailed scheduling of required resources.
- *Bottleneck analysis* — the capacity analysis level of scheduling determines the availability of machines to produce an order. It outputs a set of constraints on when each operation for an order should be performed so as to avoid unnecessary waiting, and tardiness.
- *Detailed scheduling* — the detailed level of scheduling provides complete scheduling of all resources required to produce an order. It takes into account all relevant constraints, both model and user-defined. As it constructs a schedule, it tests to see how well the schedule satisfies the known constraints. If important constraints cannot be satisfied, they will be relaxed. This level searches for a constraint-satisfying schedule.

The remainder of this section presents an overview of the ISIS approach to automatic job shop scheduling.

In constructing a job shop schedule, ISIS conducts a hierarchical, constraint-directed search in the space of all possible schedules. The different levels of the search provide multiple abstractions of the scheduling problem, each a function of the specific types of constraints that are considered at that level. Control generally flows in a top down fashion, and communication between levels is accomplished via the exchange of constraints. Processing at any given level proceeds in three phases: pre-analysis, search, and post-analysis [Figure 4.1]. The pre-analysis phase determines the bounds of the level's search space, the search phase performs the actual problem solving in this space, and the post-analysis phase assesses the quality of the results produced by the search. If deemed acceptable during post-analysis, the search results are codified as constraints for use at the next lower level of the search. Alternatively, the rejection of search results during post-analysis may lead to an alteration of the search space at

SEARCH MANAGER

pre-analysis:	search:	post-analysis:
-operator selection -constraint binding	-beam	-selection -intra-level diagnosis -inter-level diagnosis

Figure 4.1: Three phases of a level's processing

the current or a higher level (through the relaxation of one or more constraints), and the subsequent transfer of control back to the affected level.

Four search levels are organized within this framework to construct a job shop schedule. Level one selects an order to be scheduled according to a prioritization algorithm based on the category of the order, and its due date. Level two then performs a capacity analysis of the plant to determine the availability of the machines required by the selected order. Level three performs a detailed scheduling of all resources necessary to produce the order. Finally, level four selects and assigns reservations for resources required in the schedule. The following subsections consider this search architecture in more detail.

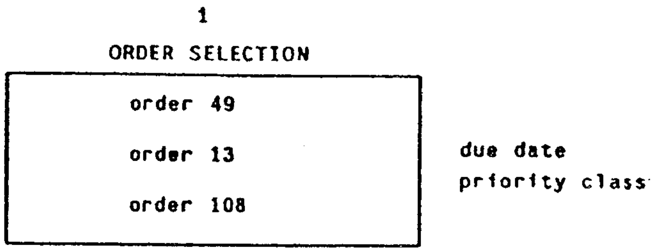

Figure 4.2: Order selection

4.1. Level one: order selection

Order selection [Figure 4.2] establishes the system's global strategy for integrating unscheduled orders into the existing job shop schedule (or loading the shop if no orders have yet been scheduled). The orders of interest at this level fall into two categories: orders that have been newly received at the plant and previously scheduled orders whose schedules have been subsequently invalidated. The invalidation of an order's schedule may occur in response to changes in the status of the plant (e.g. machine breakdowns), changes to the order's description, or decisions imposed by the user. Level one determines the current set of orders to be scheduled and prioritizes them. The scheduling priority assigned to a given order in the set is determined by its priority class (e.g. forced outage), and the closeness of its due date. Orders are then scheduled one at a time in priority order.

4.2. Level two: capacity-based scheduling

Once an order is selected, ISIS analyzes the machine capacity of the plant available to produce the order. The purpose of the capacity analysis is to detect bottlenecks in the plant so that scheduling decisions at level three may be modified (i.e., operations scheduled early enough) to satisfy time constraints such as due date, start date, and work-in-process.

Capacity based scheduling of an order selected by level one proceeds by applying a critical path method (CPM) analysis to the operations involved in the production of the order [Figure 4.3]. By considering estimates of the durations of these operations, the resource reservations previously generated by the detailed scheduling of other orders, and the order's start and due date, this analysis determines the earliest start time and latest finish time for each operation of the selected order. The times generated are then codified as operation time bound constraints and passed to level three. These constraints restrict the start and end times of operations that are subsequently generated during detailed scheduling[1].

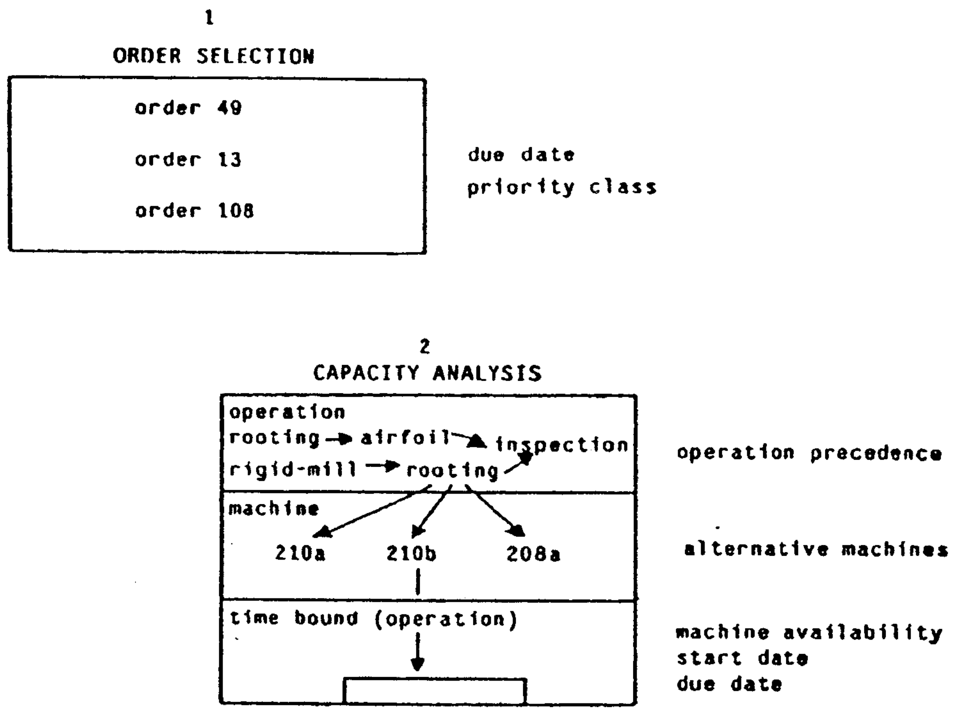

Figure 4.3: Capacity analysis level

4.3. Level three: resource scheduling

Level three [Figure 4.4] extends the scheduling of level two by considering more detailed information about operation resource requirements in addition to machines, and additional constraints such as preferences, physical constraints, etc. 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 [13] 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.

[1] *The net effect of the interpretation of these constraints is to modify the evaluation function for the search algorithm used at level three.*

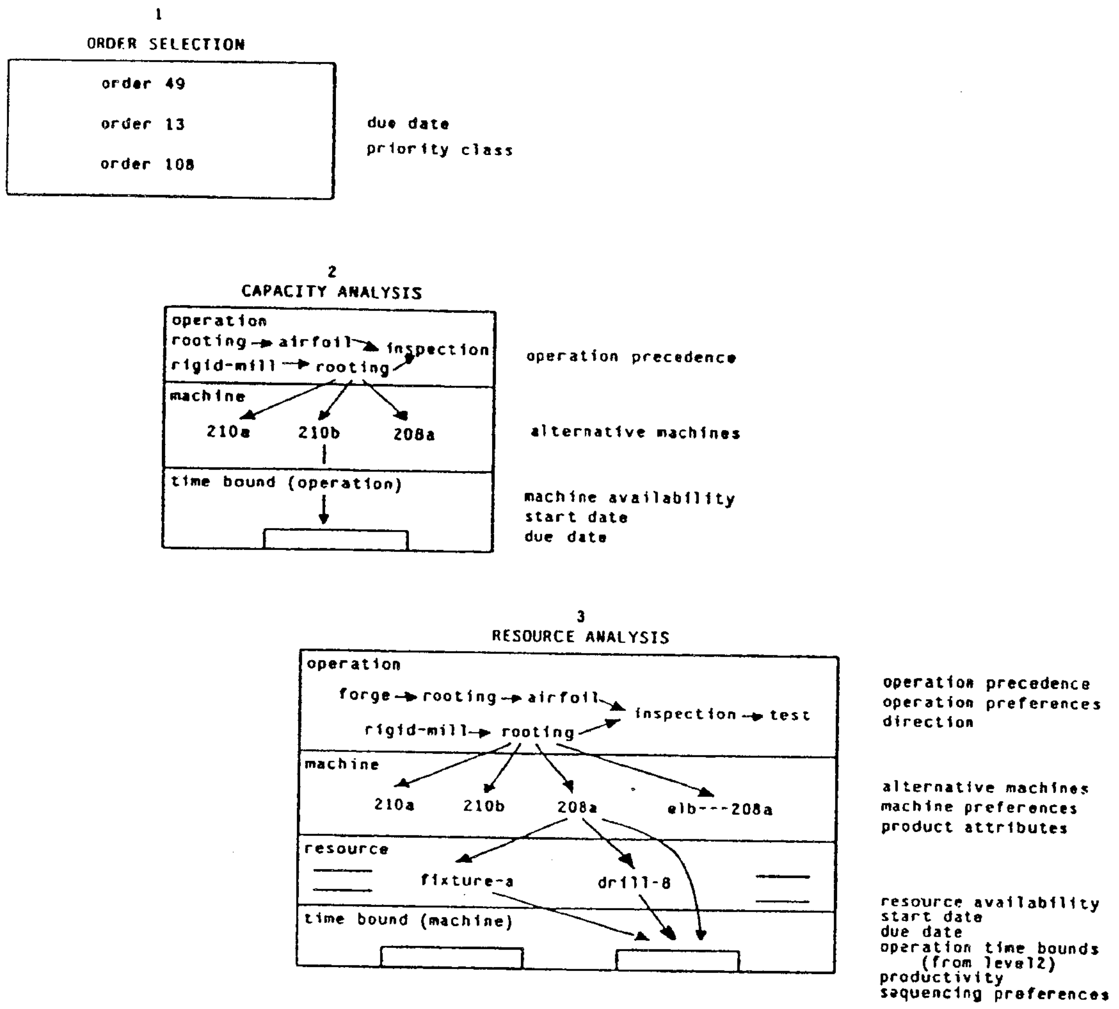

Figure 4.4: Resource scheduling level

Depending on the results of pre-search analysis, the search proceeds either forward or backward through the set of allowable routings for the order. An operator that generates states representing 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 particular resources and/or an execution time to the operation.[2] After any given operator is applied, each newly generated state is *rated* by the set of constraints found to be relevant to the state and its ancestors. This set is determined by a resolution process that extracts the constraints attached to each object specified in the state (e.g. machine, tool, order, etc.), and filters them according to the *relevance* criteria described in Section

[2] *A variety of alternatives exist for each type of operator. For example, two operators have been tested for choosing the execution time of an operation [see Section 5]. The 'eager reserver' operator chooses the earliest possible reservation for the operation's required resources, and the 'wait and see' operator tentatively reserves as much time as available, leaving the final decision to level four.*

3.2. Each constraint in the resolved set assigns a utility between 0 and 2 to the state, where 0 signifies that the state is not admissible, 1 signifies indifference, and 2 signifies maximum support. These utilities are then weighted by the importance of the assigning constraints and averaged to produce the overall rating of the state. This rating, which reflects the degree to which the partial schedule including this state satisfies the relevant constraints, serves to focus the search on subsequent iterations.

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 a type of scheduling error and the appropriate repair. Inter-level repair may result in the re-instantiation of the level's search. Pre-analysis is performed again to alter the set of operators and constraints for re-scheduling the order. Inter-level repair is initiated if 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 can be chosen.

In attempting to find a solution at this level that best satisfies the constraints, ISIS provides two approaches to the constraint relaxation:

- *Generative relaxation.* Constraints are relaxed in a generative fashion during the heuristic search. The search operators generate states where each state represents an alternative relaxation of one or more constraints.
- *Analytic relaxation.* A rule-based system analyzes an order during pre-analysis to determine the relative importance of constraints, and, in turn, which should be relaxed. Another set of rules perform a post-search analysis to determine whether the schedule is reasonable, and if not, what other constraints should be strengthened or relaxed.

4.4. Level four: reservation selection

The schedule generated by the detailed scheduling level, and passed to level four of the search, is in near final form. A specific process routing has been selected for the order under consideration, resources have been selected for each operation in the routing, and resource time bound constraints have been associated with each selected resource. Level four [Figure 4.5] finalizes the order's schedule by establishing reservations for each required resource in the schedule. Working within the resource time bound constraints provided by detailed scheduling, local optimizations are performed to minimize the order's work in process time. The resulting resource reservations are added to the existing shop schedule and act as additional constraints for use in the scheduling of subsequent orders.

5. Experimental results

Experiments have been conducted with several versions of the ISIS scheduling system, all based on a portion of the turbine component plant defined by the human plant scheduler. In each experiment, an empty job shop was loaded with a representative set of eighty-five blade orders spanning a period of two years. The various types of constraint knowledge influencing the development of schedules in these experiments included:

- alternative operations,
- alternative machines,
- due dates,
- start dates,
- operation time bounds,
- order priority classification (with orders

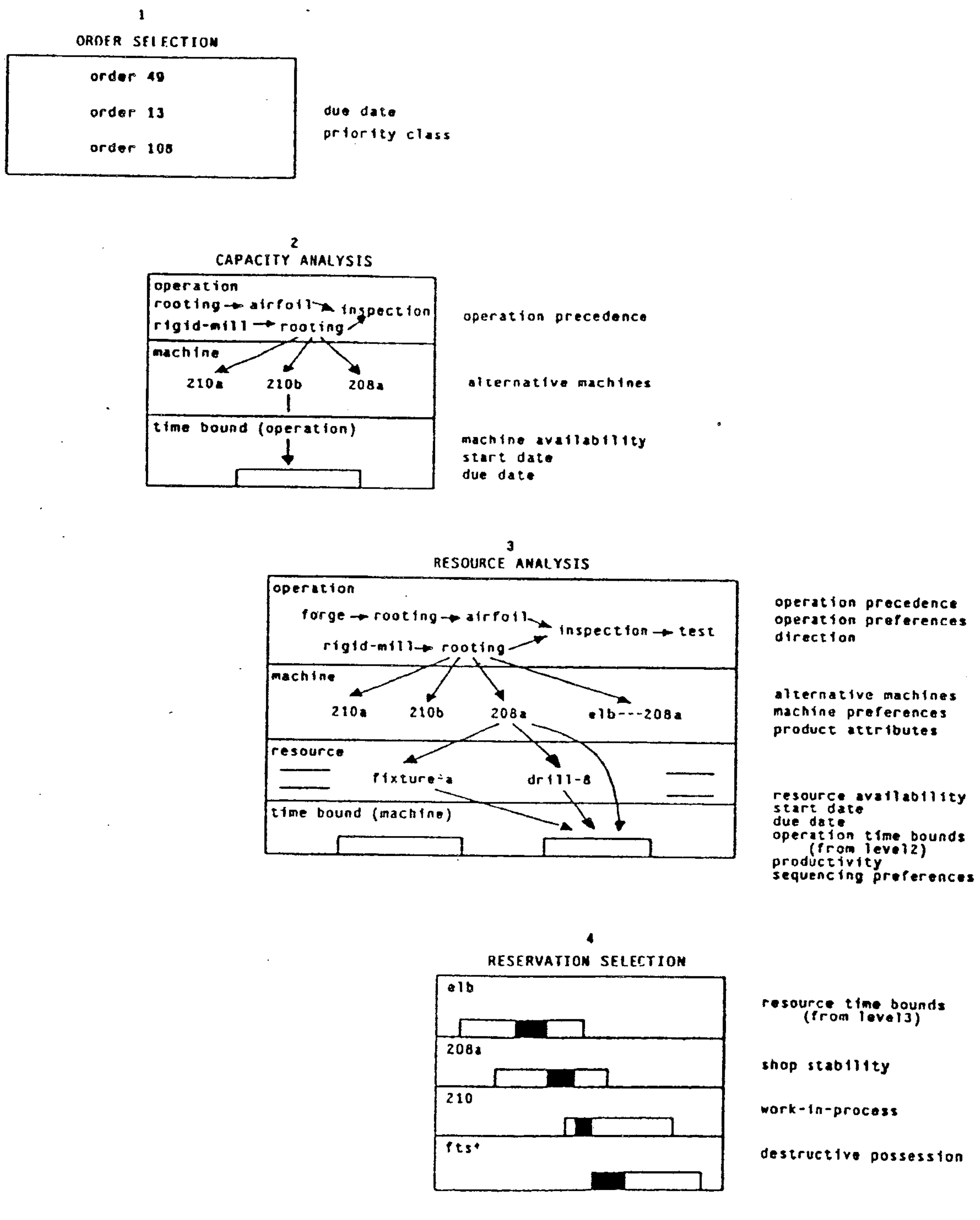

Figure 4.5: Reservation selection level

falling into four priority classes),
- work-in-process restrictions,
- sequencing constraints to reduce set-up time,
- machine constraints on product form and length,
- resource availability, and
- shop stability (minimizing pre-emption).

To date, thirteen experiments have been performed. These experiments have explored the effects of alternative constraints, alternative search operators, and the utility of the hierarchical search architecture. In this section we will examine the results obtained in two selected experiments, which serve to underscore the advantages of the hierarchical search architecture. A detailed discussion of all experiments may be found in [2].

To provide a benchmark for comparison, the initial version of ISIS tested was non-hierarchical, employing only the detailed (beam search) level of scheduling. Assignment of reservation times in this experiment was handled by the eager reserver. The gantt chart[3] shown in Figure 5.1 depicts the schedule that was generated by this version of the system. The schedule is a poor one; sixty-five of the eighty-five 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[4] of 857.4 days. The reason for these results stems 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 in Figure 5.1) 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.

The version of ISIS producing the best results in these experiments was the hierarchical system described in Section 4 employing the wait-and-see reserver. The schedule generated in this experiment is shown in Figure 5.2. The global perspective provided by the capacity based level of scheduling led to a considerable improvement in performance, evidenced most dramatically in 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

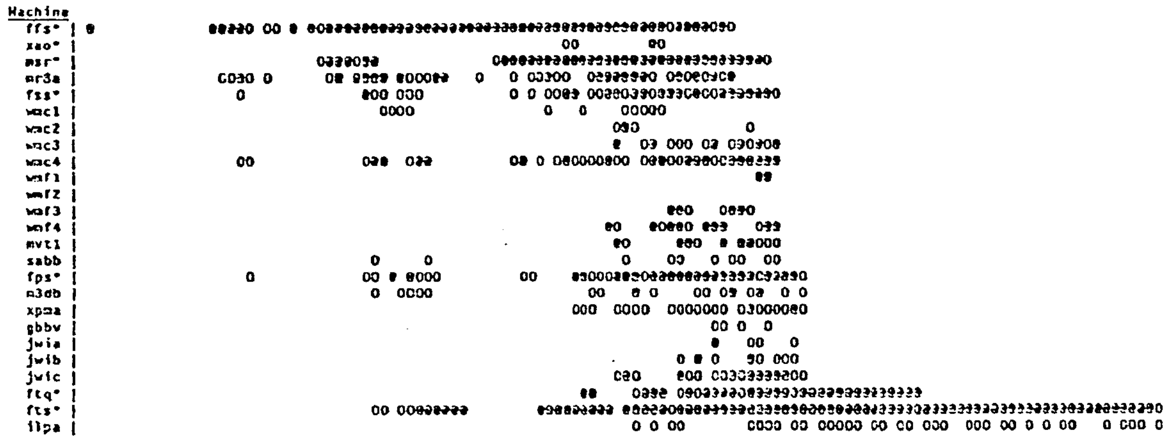

Figure 5.1: Version 1 gantt chart

[3] *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 used for less than 50% of its capacity. If the position contains a '@', then over 50% of its capacity is used. Machines that are encountered earlier in the process routings appear closer to the top of the chart.*

[4] *Makespan is the time taken to complete all orders*

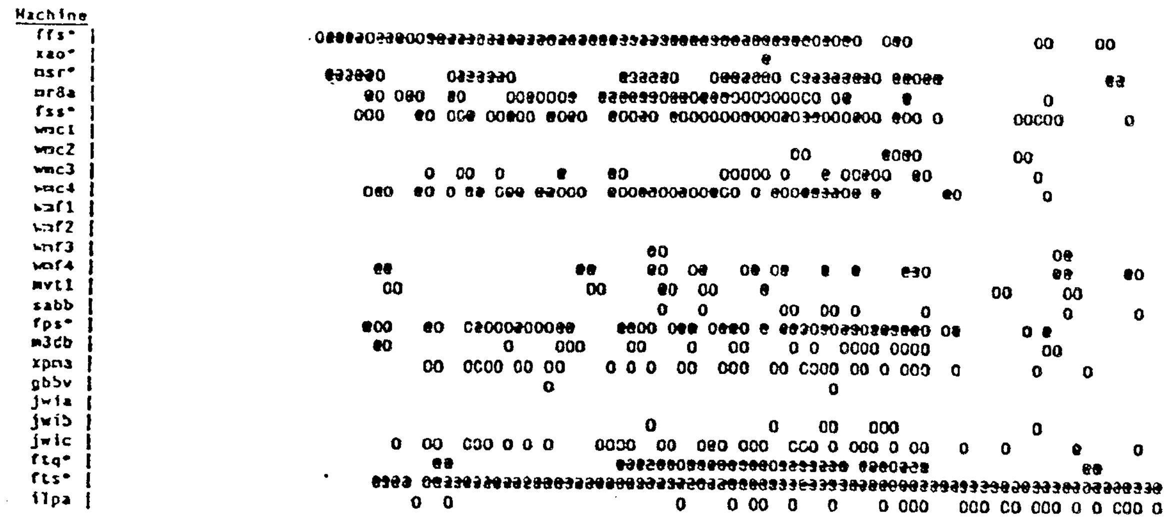

Figure 5.2: Version 7 gantt chart

fourteen. Moreover, a much lower average work-in-process time of 186.73 days was achieved, resulting in an overall make-span 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. While these results are encouraging, further testing of ISIS continues.

6. Interactive scheduling

The discussion of the previous two sections centered on the automatic generation of job shop schedules via constraint-directed search. As mentioned at the outset, ISIS also provides the user with the 'hands on' capability to construct and alter schedules interactively. In this capacity, ISIS plays the role of an intelligent assistant, using its constant knowledge to maintain the consistency of the schedule under development and identify scheduling decisions that result in poorly satisfied constraints. Capabilities provided by the interactive scheduling subsystem include:

- *Lotting* — ISIS provides the user with an interactive lotting facility for searching and examining orders, and grouping them into lots.
- *Resource scheduling* — resources may be scheduled by reserving them for use in a particular operation at user specified time. Such resources are noted both with the resource and the reserver (lot or order).
- *Hierarchical scheduling* — the user may construct schedules at different levels of abstraction, and ISIS will automatically fill in the other levels. For example, the scheduler may schedule only the critical facilities, and ISIS will complete the schedule at the detailed lineup level.
- *Overlap flagging* — if user defined reservations for resources result in conflicting assignments, ISIS will inform the scheduler, and may automatically shift other reservations.
- *Constraint checking* — whenever a reservation for a resource is made, ISIS will check all relevant constraints, and inform the user as to their satisfaction. For example, if the reserved machine cannot be used for that sized product, the user will be informed at the time the reservation is made.

This section considers the strategy employed by ISIS to provide these capabilities.

As indicated above, ISIS allows the user to interactively construct schedules at various levels of abstraction. This is accomplished by exploiting a hierarchical model of the process routings associated with an order, and maintaining reservations at all levels of abstraction. Consider a decision by the user to create a new reservation for a given order. This reservation creates a need for corresponding reservations for each of the lower-level sub-operations related to the operation in question. These are established by determining the duration of each sub-operation and scaling the resulting intervals to the interval of time designated by the user. The reservations of temporally related operations residing at higher levels are also adjusted (and created if necessary) to reflect the presence of the newly imposed lower level reservations. A decision by the user to remove a reservation is propagated to other levels in a similar fashion.

The imposition of a new reservation, and the resulting propagation of effects, may introduce conflicts into the partially developed schedule. Specifically, conflicts may arise due to 1] a violation of the temporal constraints associated with previous and/or subsequent operations in the process routing (detectable at the level of the imposition), 2] contention for the same resource by different orders (detectable at the level where actual resources are involved), or 3] the scheduling of operations belonging to mutually exclusive process routings (detectable as the imposition is propagated upward). The resolution of such conflicts involves the invalidation of one of the offending reservations. Currently, all conflicts are resolved in favor of the more recently created reservation, although other strategies (e.g. an authority model) could be straightforwardly applied. The user is informed of the schedule changes that have been made. ISIS also checks all other constraints relevant to the scheduling decision imposed by the user, and signals the user as to their satisfaction or violation.

7. Summary of ISIS features

The ISIS system has been designed to provide complete facilities for practical use in job shop production management and control. Moreover, ISIS has been designed and implemented so that its functions are independent of the particular plant under consideration. This section summarizes the additional capabilities of the current version.

7.1. Model perusal

ISIS provides full interactive model perusal and editing from multiple users in parallel. The user may alter the model of the plant from any terminal. Perusal is provided in a number of forms including menus, a simple subset of English, graphic displays, and a logic programming-based question-answering system.

7.2. Reactive plant monitoring

ISIS provides the user with interfaces to update the status of all orders and resources. If the new status does not coincide with its schedule, then ISIS will reschedule only the affected resources. For example, if a machine breaks down, then all affected orders are rescheduled. Hence, ISIS can be used to provide real-time reactive scheduling of plants. It can also be extended to connect to an online data gathering system on the plant floor, providing real-time updating of plant status.

7.3. Generative process planning

One of ISIS's features is that its constraint representation allows it to perform a subset of generative process planning. Currently, ISIS has knowledge of a product's basic physical characteristics, and can choose machines based on them. These constraints can be extended to include geometric information.

7.4. Resource planning

Since ISIS schedules all specified resources, the user is informed of the resource requirements needed to satisfy the production. Constraints on the use of resources (e.g., machines, tools, personnel, etc.) may be specified and used to guide detailed scheduling. This enables departments such as advance planning to specify resource use constraints directly to ISIS.

7.5. Accessibility: flexible interfaces

The ISIS interface is menu/window-based. The user is presented with multiple windows of information on the terminal screen, plus a menu of commands to choose from. The menu system provides a network of displays ranging from order entry/update to interactive lotting to report generation. Reports may be printed directly on the screen or directed to an attached printer. A device independent color graphics display system is also available to ISIS. It is currently used to view a blueprint-like display of the plant, and to zoom in on work areas and machines. It can be used to display the status of the plant during operation. Finally, ISIS provides a simple natural language interface for use in perusing the plant model.

ISIS has been designed to be a multi-user system. Its model can be shared amongst multiple programs. Hence, the model may be perused and altered from multiple locations in the plant, allowing departments to get the information they need to perform their tasks, and to provide information directly to ISIS. ISIS is being extended to determine who the user is and restrict access and alteration capabilities depending on their function.

7.6. Simulation

ISIS is a component of the larger Intelligent Management System Project [10] underway at the CMU Robotics Institute. Hence, it can use other functions available in the project. For example, KBS, a knowledge-based simulator [15], can interpret the organization model directly to perform simulations. Since ISIS already has a model of the plant, KBS can use that model to perform simulations. All the user has to do is modify the model to reflect environment they wish to simulate.

8. Concluding remarks

The ability of ISIS to construct realistic job shop production schedules is due to a number of representational and search techniques for reasoning with constraints. These are summarized in Figure 8.1. In adopting these techniques, the ISIS scheduling system provides, for the first time, a general methodology for incorporating the wide variety of constraints present in the job shop scheduling domain for the automatic construction of a schedule. The robustness of the constraint representation employed makes possible the introduction of almost any constraint that is deemed appropriate by a user of the system. The attention paid to the relaxations, interactions, and relevance of constraints allows constraint knowledge to be effectively applied to control the combinatorics of the underlying search space. Constraint knowledge may be used to bound the solution space, generate and discriminate among alternative solutions according to the relative importance of various constraints, communicate information between various levels of search, and diagnose unsatisfactory solutions that are proposed. In addition, the constraint representation provides the basis for a flexible interactive scheduling facility.

Work is currently underway on the next iteration of ISIS where the emphasis is on giving constraints an even more active role in the reasoning process. Specifically, we envision a system architecture that conducts a more opportunistic form of search, focusing initially on the most critical (i.e. the most highly constrained) decisions that have to be made.

Representational Techniques:

- **Specification:** explicitly specify constraints as distinct schemata in the representation, attached as meta-information to the domain model.
- **Relaxation:** specify the full set of alternative values (or relaxations) for a given constraint.
- **Utility:** associate a particular utility with each alternative relaxation of a constraint to indicate the preferences that exist amongst them.
- **Context:** indicate the contextual applicability of a constraint implicitly by attaching it to portion of the domain model that it constraints, and explicitly by specifying the precise conditions under which it should apply.
- **Importance:** specify the relative importance of satisfying various constraints.
- **Interdependence:** relate constraints to one another according to the interdependencies that exist amongst them.

Search Techniques:

- **Definition:** derive the operators that define the search space from the constraints and their relaxations.
- **Hierarchy:** use the interdependencies among constraints to stratify the search into successive levels in which more and more constraints are considered. Independent constraints appear at the top, dependent constraints at lower levels.
- **Bounding:** select only a subset of the available search operators on the basis of the importance of their corresponding constraints and the utility of their relaxations.
- **Resolution:** dynamically resolve the set of constraints that are relevant to particular decisions (states).
- **Focus:** use a search state evaluation scheme that reflects the degree of satisfaction of the resolved constraints (i.e. a function of the utilities assigned) to focus attention during the search.
- **Diagnosis:** measure how well constraints are satisfied, and weight them according to their importance to assess the quality of the search results and detect poor solutions.
- **Repair:** exploit the interdependencies among constraints for guidance in repairing poor schedules.

Figure 8.1: Representational and search techniques for reasoning with constraints

9. Acknowledgements

This work could not have been completed without the contributions of the ISIS project members which include: Brad Allen, Ranjan Chak, Peng Si Ow, Gary Strohm and Doug Zimmerman. In addition, Ari Vepsalainen and Steve Miller contributed to the analysis of the scheduling problem at the turbine component plant. Tom Morton, Ari, Ram Rachamadugu, have contributed to the definition of constraint knowledge, and have evaluated critically our progress from a operations management perspective. Our expert scheduler, Bob Baugh, has contributed to the design of the interactive system interface, and has continually made us aware of the reality of the problem. Raj Reddy, Jose Isasi, Dwight Mize, and Bob Baugh have provided continued support during the ups and downs of this project.

10. References

[1] M.S. Fox, B.P. Allen and G.A. Strohm, 'Job Shop Scheduling: An Investigation in Constraint-Directed Reasoning,' *Proceedings of the 2nd National Conference on Artificial Intelligence*, American Association of Artificial Intelligence, August 1982, pp. 155–158.

[2] M.S. Fox, 'Constraint-Directed Search: A Case Study of Job Shop Scheduling,' PhD thesis, Carnegie-Mellon University, 1983.

[3] M.S. Fox, S.F. Smith, B.P. Allen, G.A. Strohm and F.C. Wimberly, 'ISIS: A Constraint-Directed Reasoning Approach to Job Shop Scheduling,' *Proceedings IEEE Conference on Trends and Applications 83*, Gaithersberg, Maryland, May 1983.

[4] L. Steels, 'Constraints as Consultants,' AI Memo 14, Schlumberger-Doll, December 1981.

[5] M. Stefik, 'Planning with Constraints (MOLGEN: Part 1),' *Artificial Intelligence,* **16**, 1981, pp. 111–140.

[6] G.J. Sussman and G.L. Steele, Jr. 'CONSTRAINTS — A Language for Expressing Almost-Hierarchical Descriptions,' *Artificial Intelligence,* **14**, 1980, pp. 1–40.

[7] D. Waltz, 'Understanding Line Drawings of Scenes with Shadows,' P.H. Winston (editor), *The Psychology of Computer Vision*. McGraw-Hill, New York, 1975.

[8] M.S. Fox, 'On Inheritance in Knowledge Representation,' *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*. Tokyo, 1979.

[9] J.M. Wright, M.S. Fox and D. Adam, *SRL/1.5 User Manual*, Technical Report, Robotics Institute, Carnegie-Mellon University, 1984.

[10] M.S. Fox, *The Intelligent Management System: An Overview,* Technical Report CMU-RI-TR-81-4, Robotics Institute, Carnegie-Mellon University, August, 1981.

[11] S.F. Smith, *Exploiting Temporal Knowledge to Organize Constraints*, Technical Report CMU-RI-TR-83-12, Robotics Institute, Carnegie-Mellon University, 1983.

[12] C. Engleman, E. Scarl and C. Berg, 'Interactive Frame Instantiation,' *Proceedings of the 1st National Conference on Artificial Intelligence*, American Association of Artificial Intelligence, 1980, pp. 184–186.

[13] B. Lowerre, *The HARPY Speech Recognition System*, PhD thesis, Computer Science Department, Carnegie-Mellon University, 1976.

[14] B. Allen and J.M. Wright, 'Integrating Logic Programs and Schemata,' *Proceedings of the International Joint Conference on Artificial Intelligence*, Karlsruhe, Germany, Aug. 1983.

[15] Y.V. Reddy and M.S. Fox, *KBS: An Artificial Intelligence Approach to Flexible Simulation*, Technical Report CMU-RI-TR-82-1, Robotics Institute, Carnegie-Mellon University, 1982.

About the authors

Mark S. Fox

Mark Fox heads the Intelligent Systems Laboratory of the Robotics Institute at Carnegie-Mellon University and is an Assistant Professor of Management Science. His research interests span both computer and management science, including artificial intelligence, man–machine communication, databases, software system organisation, graphics, operations management, and organisation theory. He is the principal investigator of the Intelligent Management Systems project which performs research in the design and construction of AI-based systems to automate the management and control functions of factories. Dr. Fox received a B.Sc. in Computer Science from the University of Toronto in 1975, and his Ph.D. in Computer Science from Carnegie-Mellon University in 1983. He was twice awarded a National Research Council of Canada Post-Graduate Scholarship.

Stephen F. Smith

Stephen Smith is a Research Scientist in the Intelligent Systems Laboratory of the Robotics Institute at Carnegie-Mellon University. He received his B.S. degree (1975) in Mathematics from Westminster College and his M.S. (1977) and Ph.D. (1980) degrees in Computer Science from the University of Pittsburgh. Prior to joining the faculty of the Robotics Institute in 1982, he was an Assistant Professor of Computer Science at the University of Southern Maine. Dr. Smith's research interests include hierarchical and distributed problem solving, knowledge-based systems, machine learning, and AI applications to industrial problems. He has been involved with the ISIS project for the past two years. Dr. Smith is a member of the American Association of Artificial Intelligence, the ACM, and the IEEE Computer Society.

only
CONNECT
. . . and your micro
brings you the
World!

Intelligent Terminal Software
for Microcomputers

For more information on CONNECT which costs £195 contact
Learned Information (Europe) Limited,
Besselsleigh Road, Abingdon, Oxford OX13 6LG.
Telephone: (0865) 730275.