ISIS: A Constraint-Directed Reasoning Approach to Job Shop Scheduling

System Summary

Mark S.Fox, Bradley P.Allen,
Stephen F.Smith, Gary A.Strohm

CMU-RI-TR-83-8

Intelligent Systems Laboratory
The Robotics Institute
Camegie-Mellon University
Pittsburgh, PA 15213

June 21, 1983

ISIS: A CONSTRAINT-DIRECTED REASONING APPROACH TO JOB SHOP SCHEDULING
System Summary

Mark S. Fox, Bradley P. Alien, Stephen F. Smith, Gary A. Strohrn

intelligent Systems Laboratory
The Robotics Institute
Carnegie-Mellon University
Pittsburgh, PA 15213

21 June 1983

Abstract: Analysis cf the job shop scheduling domain has indicated that the crux of the scheduling problem is the determination and satisfaction of a large number and variety of constraints. Schedules are influenced by such diverse factors as due date requirements, cost restrictions, production levels, machine capabilities, operation precedences, 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 utilized 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.

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

A version of this paper appeared in the *Proceedings of IEEE Conference on Trends and Applications 83,* National Bureau of Standards, Gaithersburg, Maryland, May 1983.

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 factors as due date requirements, cost restrictions, production levels, machine capabilities, operation precedences, resource requirements, and resource availability. Most current approaches to automatic scheduling incorporate only a fraction of this knowledge. This leads to 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 factory state, leaving detailed scheduling to the shop-floor supervisor. Automatic scheduling is then reduced to a weekly or monthly runs whose outputs provide guidance in determining future loadings. By adopting a constraint-directed reasoning approach to the scheduling problem, it is possible to include all relevant scheduling knowledge in the schedule generation and selection process. Within this paradigm, the problem of scheduling orders in a job shop involves such issues as: extending knowledge representation techniques to include the variety of constraints found in the scheduling domain, integrating constraints into the search process (in particular, determining how to use constraints to bound the generation and focus the selection of alternative solutions), relaxing constraints when conflict occurs, and analyzing the interaction between constraints to diagnose poor solutions. Using constraint-directed reasoning, a *reactive* capability is added to the scheduling system. Comprehensive schedules can be constructed in response to the actual current state of the factory. This paper describes a system called ISIS which takes this latter approach to job shop scheduling.

The ISIS system has been designed to provide complete facilities for practical use in job shop production management and control. As with all knowledge based systems, the power of ISIS lies in its rich model of the job shop environment. The functionality currently supported by this model include organizational modeling, model perusal and editing, automatic scheduling, interactive scheduling, reactive plant monitoring, and simulation. We will limit the discussion here to scheduling issues only. A more complete description can be found in [5].

The remainder of the paper is organized as follows. 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 utilization 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. We turn our attention to the practical capabilities provided by the ISIS interactive scheduling subsystem in Section 5. Finally, in Section 6, 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 satisfied 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 10 orders through 5 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 TCP Job Shop Environment

A Turbine Component Plant (TCP) 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 TCP, we found that orders are not scheduled in a uniform manner. Each scheduling decision to be made entails side effects whose importance varies b)^order, and the genei§tiQn-Qt-^ Sjcheidule js anaterative 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% to 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. These constraints include 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 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 utilization. What is needed is a general methodology for utilizing 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 sch-eduling constraints to be considered and Jheir affect on the scheduling process. Our a7TalysiB'1Df the constraints present in the TCP 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 *Oj^QnJzafjqnal GgaL* 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.
- 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 may be a graph which dictates how long a drill can run at a particular speed in a particular material.

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.

A fifth category of constraint is *Preference*. 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, machine preferences, operation preferences, and queue position preferences exemplify this type of

constraint.

The following table lists the variety of constraints we have identified as well as the categories we have used to classify them.

Constraint	Org. Goal	Physical	Causal	Availability	Preference
Operation alternatives			x		
Operation Preferences					x
Machine alternatives			x		
Machine Preferences					x
Machine physical constraints		x			
Set-up times	x	x			
Queue ordering preferences					x
Queue stability	x				
Due date	x				
Work-in-process	x				
Tool requirement			x		
Material requirement			x		
Personnel requirement			x		
Resource reservations				x	
Shifts	x			x	
Down time				x	
Productivity achieved	x				
Cost	x				
Productivity goals	x				
Quality	x	x			
Inter-operation transfer times			x		

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 utilizing 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 TCP. On the other hand, Management Science research has focused on optimal results for artificial problems, or dispatch rules for meeting due dates or makespan (i.e., facility utilization). 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 is the repository of all the knowledge necessary to plan and schedule production in a job shop environment. The system is built using SRL [2, 11], a flexible knowledge representation system which allows the user to mold 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 a relation are user definable. Figure 3-1 illustrates the basic SRL construct

```
{{ 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

SRL Construct defining Op. Schema

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 [3] including simulation, diagnosis, graphics, project management, and long range planning.

The ISIS modeling system extends SRL by providing a variety of primitives for modeling manufacturing organizations. At the lowest level, the concepts of states, objects, and acts are provided as well as a set of temporal and causal primitives for relating them. These primitives provide a foundation upon which higher level primitives such as manufacturing operations, resources, products, orders, etc. are defined and related. In specifying process routings, for example, manufacturing operations are defined as acts, and relations such as NEXT-OPERATION are composed from basic temporal and causal relations. Required resources are expressed as objects, whose allocation is defined as a state of possession by a particular operation. An additional primitive capability enables the attachment of constraints to any concept defined 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 operations 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 precedence 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** and **possess-wrench**, each of which are also linked to the **milling** operation via causal (e.g. ENABLE) and temporal (e.g. OVERLAP, 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 Additional schemata are present in the representation to describe the properties of the relations themselves. With respect to Figure 3-2 the domain specific relations NEXT-OPERATION, SUB-OPERATION, AND SUB-STATE are defined in terms of more primitive domain independent relations, forming the relation hierarchy shown in Figure 3-4.

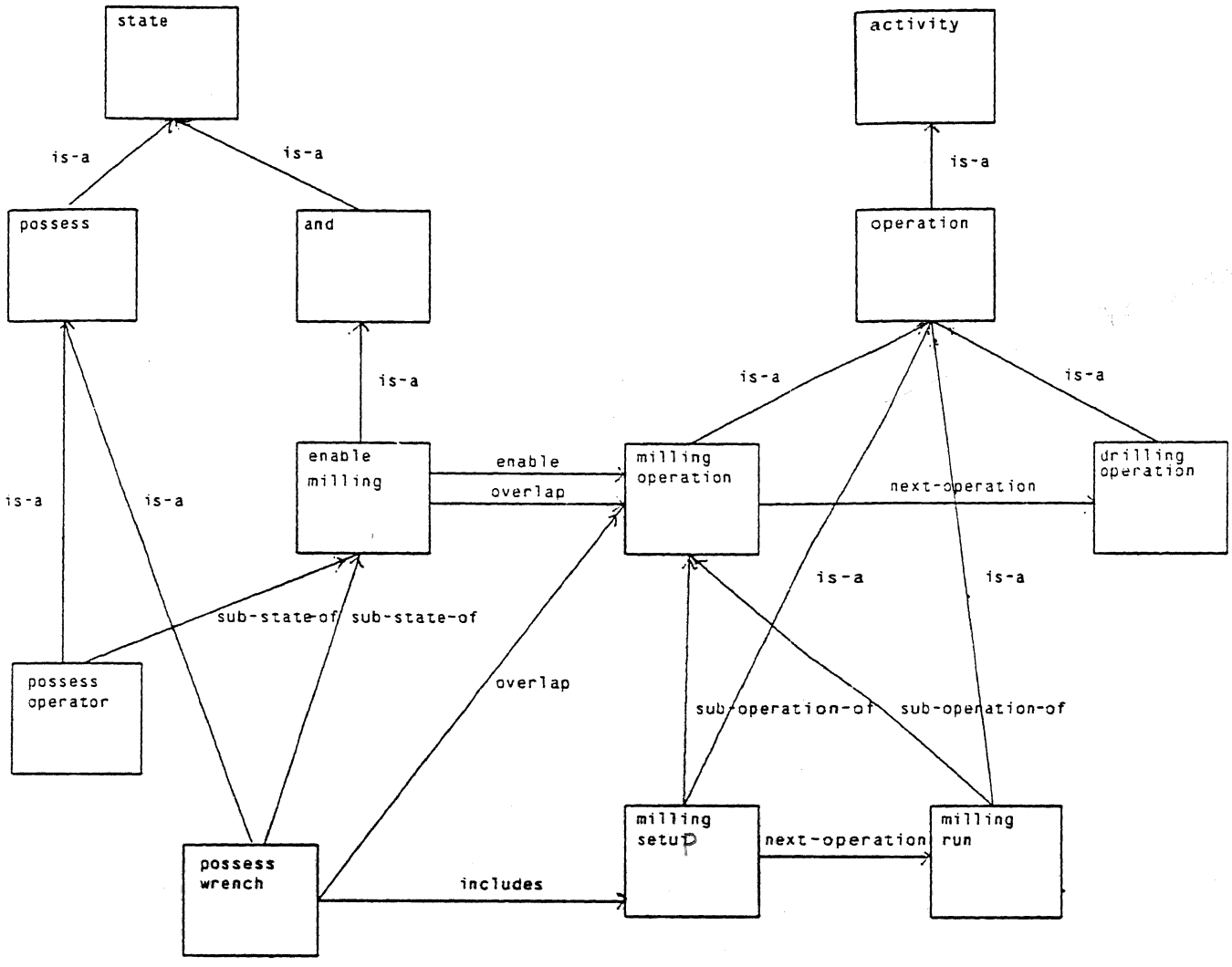


Figure 3-2: Activity Model

3.2. Constraint Representation

Given the central role of constraints in determining a job shop schedule, a major objective of our research has centered on the development of a general characterization of constraint knowledge to support constraint-directed search. As an example of constraint knowledge, consider 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, not all due dates can 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?

```
{{ 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

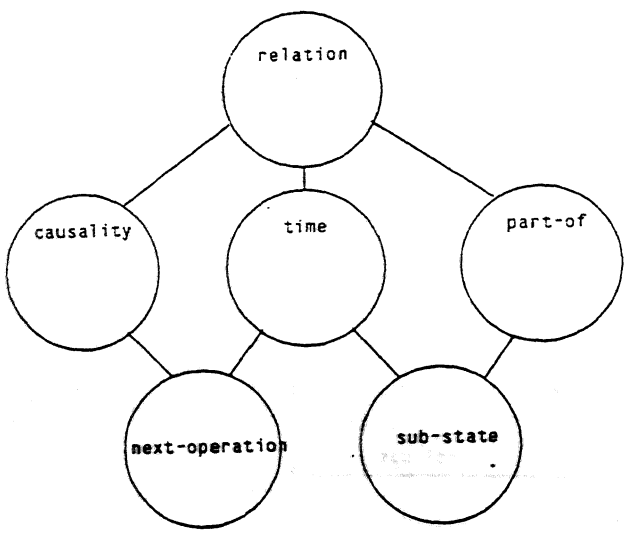


Figure 3-4: Relation Hierarchy

- what the 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 [5, 4, 7]).

One of the first issues to be faced in the representation of constraints 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 accomplisFTthe Tatter may require using faster, more expensive machines, thereby causing a conflict with the former. If the conflict cannot be solved, one or both constraints must "give ground" or 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 [8, 1] or choice sets [9, 10].

Within ISIS, this capability is provided by extending the constraint representation to include the specification of *relaxations* (i.e. alternative values) for constraints. Relaxations may be defined as either^predicates or choice sets. In the latter case, they are further distinguished as discrete or continuous. The simple specification of bounds on a variable, however, provides no means of differentiating between the values falling within these bounds. Such a capability is requi'ed by the reasoning system to effectively discriminate among the alternative partial schedules generated to resolve a given conflict. Accordingly, associated with each relaxation there is a preference measure (utility) that indicates the preferred relaxations among those available. This knowledge is also used to guide the generation of various alternatives for consideration.

A second aspect of the constraint representation is *importance.* Not all constraints are of equal importance. The due date constraint associated with a high priority order, 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 aspect of the constraint representation concerns the *interaction* of constraints. Constraints do not exist independently of one another, but rather the satisfaction of a given constraint wi!l 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 interactions are expressed as relations within the ISIS constraint representation, with an associated *sensitivity* measure indicating the extent of the interaction. Knowledge of these interactions is utilized 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 fourth characteristic of the constraint representation is constraint *obligation,* 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 obligation can be determined to a large degree by the proximity of constraints to the portion of the model currently under consideration. A flner level of discrimination is provided by associating a specific *context* of applicability with each constraint. However, our experience with factories has uncovered problems in the application of constraints solely on the basis of 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 [7]. A second problem involves inconsistencies that might arise with respect to a given constraint type. ISIS may be used by a number of departments in the factory and could result in different variants of the same constraint type being created and applied to the same object. 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 final concern is that of constraint *generation.* Constraints have numerous sources. Many may be defined by the user during the creation of the plant model. Others may be defined dynamically as the production proceeds. For example, the constraint on the mass of metal removed during an operation is dependent on the mass of the metal before the operation. Hence, this constraint is determined at the time the operation is performed. The dynamic creation and propagation of constraints is accomplished by attaching constraint generators to appropriate relations in the model.

{{ due-date
{ is-A range-constraint
IMPORTANCE:
CONTEXT: t
DOMAIN:
range: (type IS-A lot)
RELATION: due-date
CONSTRAINT:
range: (type SS-A due-date-constraint) }
PRIORITY-CLASS: }}

Figure 3-5: due-date Schema

{{ due-date-constraint
{ IS-A continuous-constraint
CONSISTENCY: exclusive
DOMAIN: dates
PIECE-WISE-LINEAR-UTIUTY: } }}

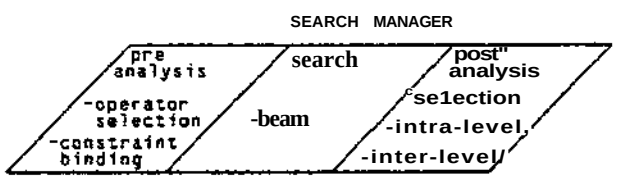
Figure 3-6: due-date-constraint Schema

An example of a constraint within the ISIS model Is a due-date (figure 3-5). It constrains the range (l.e. value) that a slot may have. In particular, it constrains the DUE-DATE slot (relation) associated with a lot schema The specific constraint that is imposed[1] on this slot is described by the due-date-constraint schema (figure 3-6), which is defined as a type of continuous-constraint. A

continuous constraint restricts the value of a slot to a particular domain, in this case the domain of dates, and a specifies a piece wise linear function for determining the utility of any particular value chosen.

4. Constraint-Directed Search

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 Xfunction 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 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.



Figu re 4-1: Three Phases of a Level's Processing

Four search levels are organized within this framework to construct a job shop schedule. Level 1 selects an order to be scheduled according to a prioritization algorithm based on the category of the order, and its due date. Level 2 then performs a capacity analysis of the plant to determine the availability of the resources required by the selected order. Level 3 performs a detailed scheduling of all resources necessary to produce the order. Finally, level 4 selects and assigns reservations for resources required in the schedule. The following subsections consider this search strategy in more detail.

4.1. Level 1: 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 1 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.

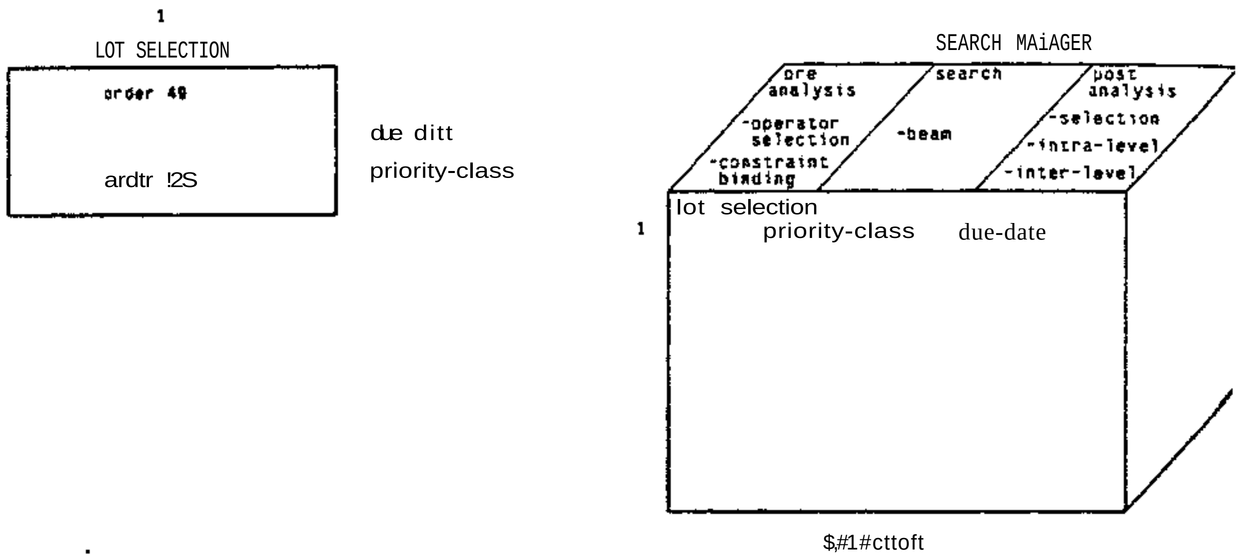


Figure 4-2: Lot Selection

4.2. Level 2: Capacity based scheduling

Capacity based scheduling of an order selected by level 1 proceeds by applying a critical path method (CPM) analysis to the operations involved in the production of the order (figure 4-2). 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 and latest finish times for each operation of the selected order. The times generated are then converted to time bound constraints and passed to level 3. These operation time bound constraints constrain the start and end times of operations that are subsequently generated during detailed scheduling.

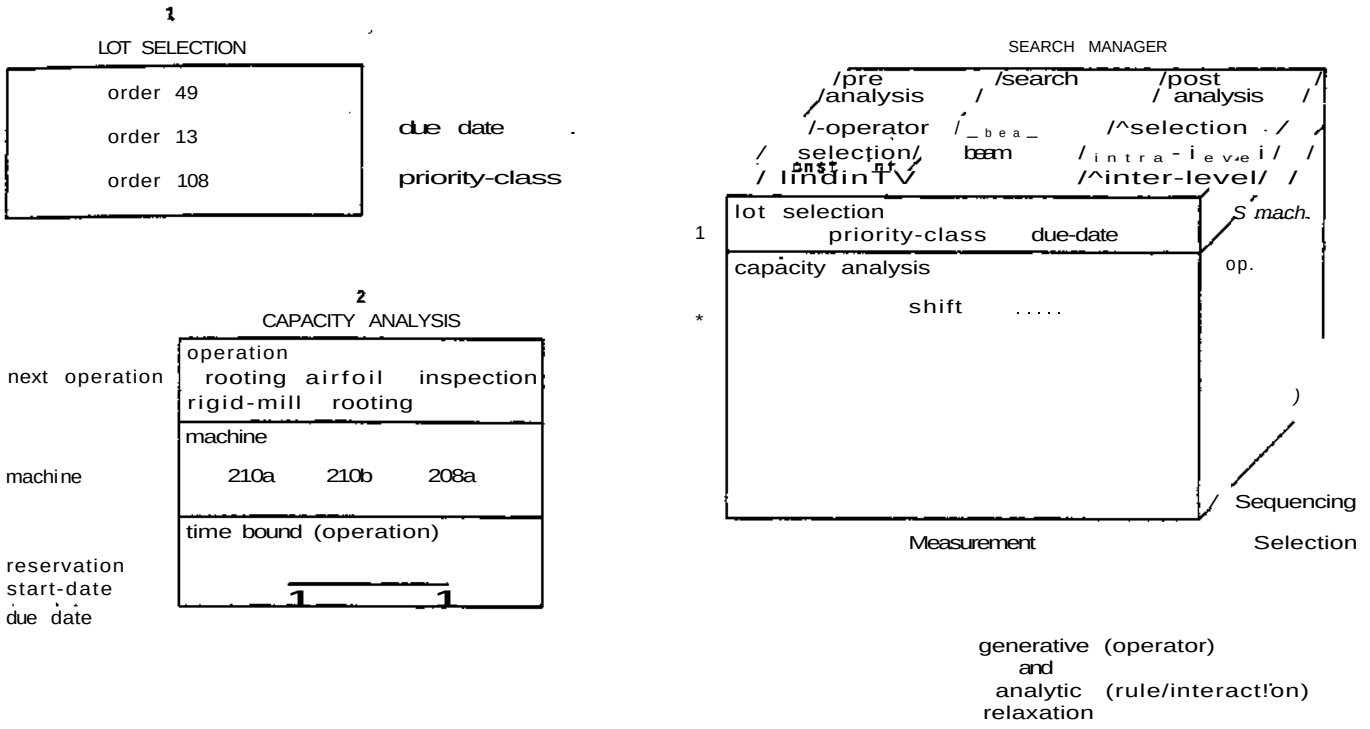


Figure 4-3: Capacity Analysis Level

4.3. Level 3: Detailed scheduling

The detailed scheduling of an order (figure 4-4) begins with a pre-search analysis. This analysis examines the constraints associated with the order to determine the scheduling direction (forward vs backward), whether any additional constraints should be created (e.g., due dates, work-in-process), and the search operators which will generate the search space. A beam search [6] is then performed using the selected search operators.

The beam search sequences the application of operators to elaborate the search space. Starting with a null schedule, alternative partial schedules are generated either forward from the start date or backward from the due date (depending on the direction determined by pre-search analysis). An operation operator generates alternative states which represent afternative operations in either the forward or backward direction. Once the operation is known for a state, other operators extend the search by creating new states which bind the resource(s) and/or *the* execution time of the operation. Thus, each application of an operator generates another "ply" in the search space. At each ply only the "n" highest rated states (see below) are selected for extension to the next ply.

The most frequently selected operators generate alternative operations, machines, and queue positions for an order in the plant. Other resources (e.g., tools, materials, etc.) are generated by other operators. With respect to each of these operator types, a variety of alternatives exist For example, two operators have been tested for choosing the execution time of an operation. The **eager reserved operator chooses the earliest possible reservation for the operation's required resources.

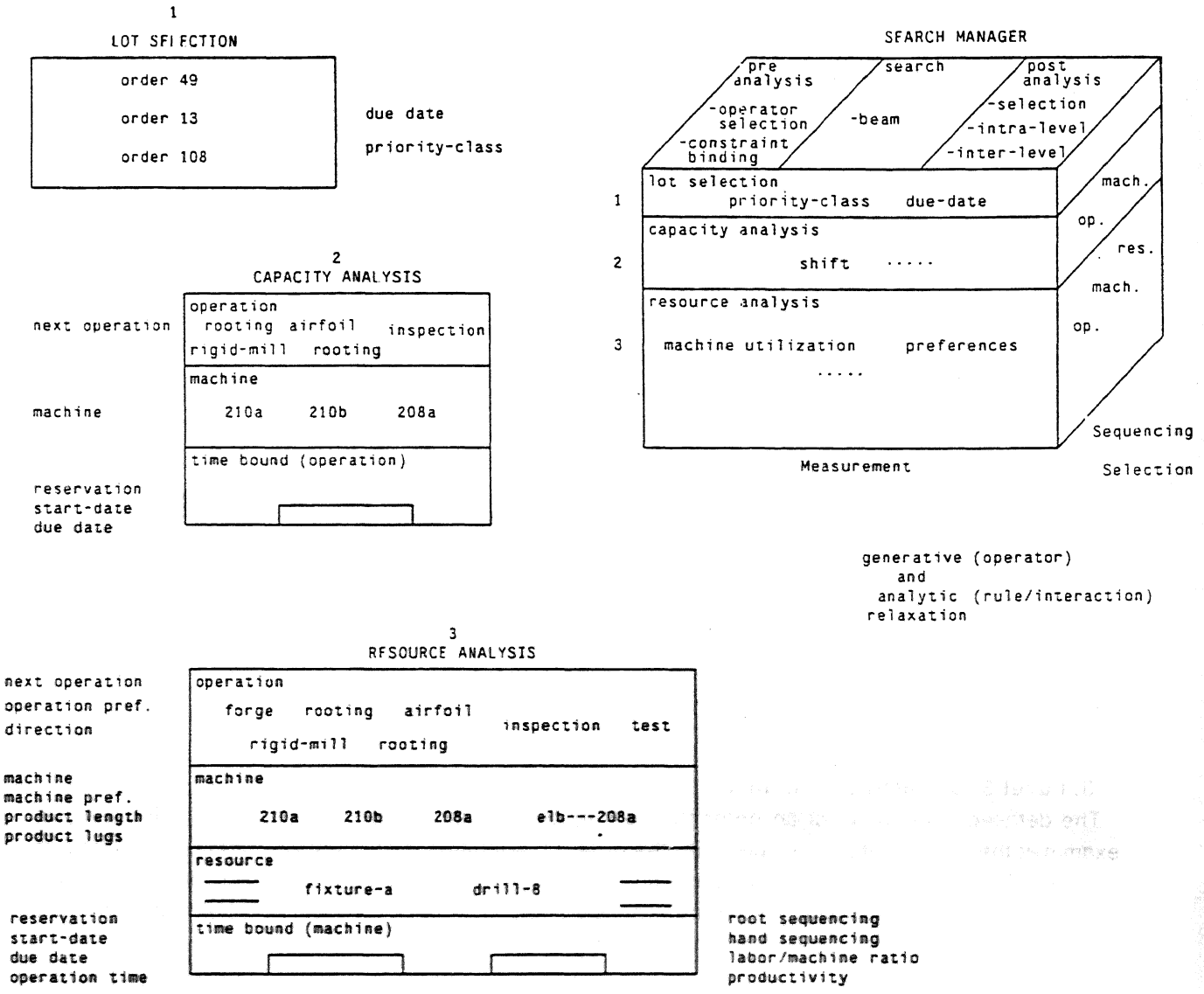


Figure 4-4: Detailed Scheduling Level

while the "wait and see" operator tentatively reserves as much time as available. Both operators generate bounds on the reservation times for each resource being considered in the operation. These resource time bound constraints are used to focus reservation selection at level 4.

Each state in the search space is rated by the set of constraints found (resolved) to be relevant to the state and its ancestors. The rating of a state can be divided into two parts: resolving what constraints should be applied to the state, and applying the constraints to the state. As the search proceeds, states are generated which vary widely in their choice of operations, machines, and queue positions. Not all constraints in the system may be relevant in rating the state (partial schedule) in question. The applicable constraints are dynamically determined, and may originate from four sources: their placement in the *plant model*, their *hierarchical imposition* by others systems such as

capacity analysis (e.g., removing a routing due to bottlenecks), their *lateral imposition* early on in the search (e.g., choosing an operation early in the routing may disqualify a later operation), and their *exogenous imposition* by the user. After the local constraint set is resolved, ISIS filters the set by evaluating each constraint's context. The context is the final test of a constraint's applicability. Only constraints with a true context form the final local constraint set.

Unlike some simple game tree searches, the path which leads to a search state is as important as the state itself. Each state in the path defines a single set of operation, machine, and queue bindings for the order, and a complete schedule is defined by the path from the initial state to the end state in the search space. Accordingly, the rating assigned to a state reflects the quality of the entire partial schedule leading to the state, rather than the single scheduling decision represented by the state. Local resolution, as defined above, collects only the constraints that bear directly on the state under consideration. To produce the partial schedule rating, ISIS also includes the constraints applied to all the states lying on the search path terminating at the current state.

In collecting the constraints applied to earlier states in the partial schedule under evaluation, ISIS distinguishes between two categories: *invariant* (e.g. operation preference, queue ordering) and *transient* (e.g. a due-date or work in process estimator). All of the invariant constraints along the path from the initial state to the current state are collected for inclusion in the rating. ISIS also gathers up all of the transient constraints, but does not retain duplications. Only the latest instantiation of a transient constraint (i.e. the one closest to the current state) is saved. These two sets of constraints are merged with those resolved locally to form the final constraint set for the state under consideration.

Upon determination of the final constraint set, each constraint is weighted to reflect the relative influence it should exert in the rating to be assigned. As indicated in Section 3.2, the importance of various constraints may be defined statically or derived dynamically according to pre-specified *scheduling goals*. Each applicable constraint then assigns a utility (i.e. its relaxation preference measure) to the state. This utility falls within the range of zero to two, with zero signifying that the state is not admissible, one signifying indifference to the state, and two signifying maximal support. The rating of a state with multiple relevant constraints is the weighted (by importance) average of the constituent constraints.

Once a set of candidate schedules has been generated, a post-search analysis examines the candidates to determine if one is acceptable. Currently, any schedule with a rating greater than one is accepted. If no acceptable schedules are found, then diagnosis is performed. First, the schedules are examined to determine a type of scheduling error. The error is then fed back to pre-analysis in order to select new operators which are used to reschedule the same order. The diagnosis of poor solutions caused by constraint satisfaction decisions made at another level 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 chosen.

At this level, ISIS provides two approaches to the relaxation of constraints:

- **Generative Relaxation.** Constraints are relaxed in a generative fashion in 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 4: Reservation Selection

The schedule generated by the detailed scheduling level, and passed to level 4 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 4 (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.

4.5. Performance results

Experiments have been conducted with several versions of the ISIS scheduling system, all based on a portion of the turbine plant defined by the human plant scheduler. In each experiment, an empty job shop was loaded with a representative set of 85 blade orders spanning a period of two years. Constraint knowledge employed in developing schedules included:

- alternative operations,
- alternative machines.
- cluedates.
- start dates.
- operator! time bounds.
- order priority classification.
- « work in process restrictions.
- queue ordering constraints to reduce setup tiem.
- machine constraints on product form and length.
- resource availability.
- shop stability (minimizing pre-emption).

This section presents the results obtained in two selected experiments. A detailed discussion of all experiments may be found in [SJ.

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 reserves The gantt chart[1] shown *in* Figure 4-6 summarizes the

[1] Each row represents a machine, and each column a week, Sf a position in the gantt chart a empty* then *the* machine does rot **have** any orders for that week<. If a position contains an "o", then il ^s utilized for Ie» *thm* 50% of its capacity, ff **the position** contains a #@*\ **then** over 50% of its capacity ss utilized, Machines **which** are **used** earlier in lineups appear closer to **the** top of **the** chart.

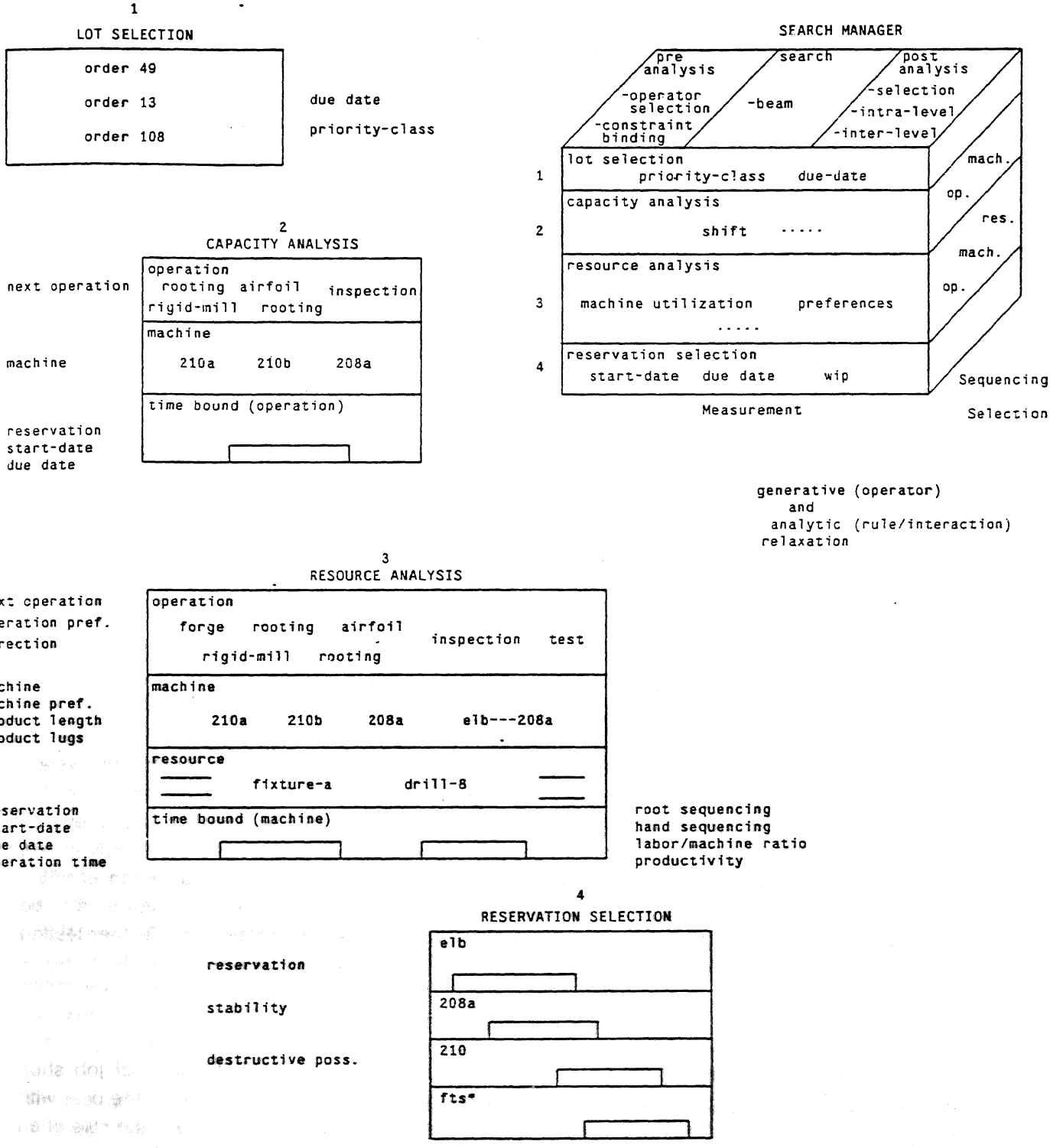


Figure 4-5: Reservation Selection Level

schedule that was generated by this version of the system. The schedule is a poor one; 68 of the 85 orders scheduled were tardy. To compound the problem, order tardiness led to high work in process times with an overall makespan of 903 days. The reason for these results stems from the inability of the beam search to anticipate the bottleneck in the final straightening area (the fts* machine on the gantt chart in Figure 4-6) 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.

Figure 4-6: Version 1 Gantt Chart

The version of ISIS producing the best results in these experiments was the hierarchical system described above employing the wait-and-see reserver. The schedule that was generated in this experiment is shown in Figure 4-7. The global perspective provided by the capacity based level of scheduling led to a considerable improvement in performance, reducing the number of tardy orders to 14. Moreover, very low work in process times were achieved with an overall makespan of 565.8 days. In this case, inadequate machine capacity in the final straightening area (fts*) appears to be the principal limitation affecting order tardiness. While these results are encouraging, further testing of ISIS is ongoing.

5. 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 capability to interactively construct and alter schedules. In this capacity, ISIS plays the role of an intelligent assistant, utilizing its constraint knowledge to maintain the consistency of the schedule under development and identify scheduling decisions that result in poorly satisfied constraints. This section examines the characteristics of ISIS's interactive scheduling capability.

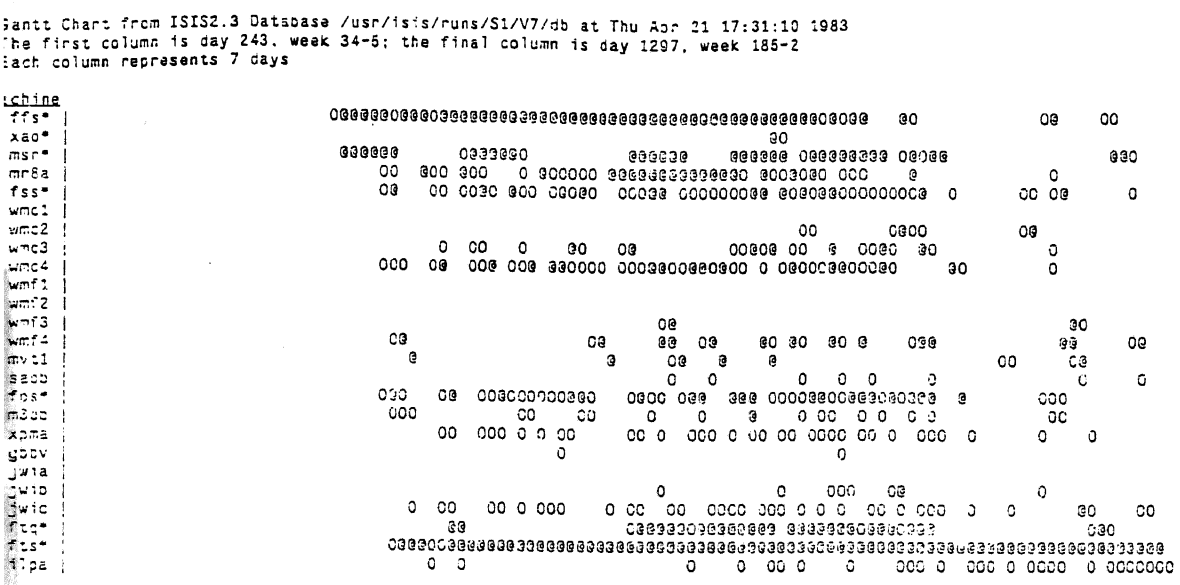


Figure 4-7: Version 5 Gantt Chart

ISIS allows the user to interactively construct schedules at various levels of abstraction. This is accomplished by employing a hierarchical model of the process routings associated with an order, and maintaining resource reservations at all levels of abstraction. The resources required by abstract operations embody abstractions of the resources required by their constituent suboperations. Consistency is maintained by propagating each scheduling decision imposed by the user to all levels. Thus, the status of a given reservation at any point in time reflects the status of the reservations supporting it at lower levels. The temporal constraints embedded in the model (e.g. suboperations occur *during* the same interval of time as the operation that abstracts them, previous operations occur *before* the current operation, etc.) serve to focus the propagation process.

Let us first consider a decision by the user to impose a new reservation. The existence of this reservation implies the existence of corresponding reservations at lower levels. Thus, if the operation involved in the user's decision is an abstraction, reservations are established for each of the abstracted suboperations. At present, this is performed by determining the duration of each suboperation and scaling the resulting intervals to the interval of time designated by the user, recursively applying the procedure in the event of a suboperation that is itself an abstraction. The reservations of temporally related operations residing at higher levels are adjusted (and created if necessary) to reflect the presence of the newly imposed lower level reservations.

The imposition of a new reservation, and the resulting propagation of effects described above, 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.

Similarly, a decision by the user to remove a reservation is propagated to other levels. If the decision involves an abstract operation, any existing reservations associated with the abstracted suboperations are also removed, and the process is recursively applied to each suboperation that is itself an abstract operation. The reservations associated with temporally related operations at higher levels are adjusted (and possibly removed) to reflect the user's decision. Any previously invalidated reservations for which a conflict no longer exists are restored and the user is informed of the action.

Decisions made by ISIS to invalidate or restore a reservation are also propagated. The techniques are analogous to those described above for the removal and establishment of a reservation respectively.

6. Concluding remarks

The ISIS scheduling system provides, for the first time, a general methodology for representing and utilizing 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 incorporation of any constraint that may be deemed relevant by a user of the system. The attention paid to relaxations, interactions, and obligations 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 are interested in relaxing the beam search and employing constraints in a more procedural fashion during the search. We envision a distributed system architecture in which constraints, acting as independent knowledge sources, cooperate opportunistically to produce a schedule. Such an organization, in turn, will provide the necessary framework for a totally integrated interactive/automatic scheduling system.

7. Acknowledgements

Ari Vepsalainen and Steve Miller contributed to to the analysis of the scheduling problem at the turbine component plant. Tom Morton, Ari, Ram Rachamadugu, and Peng Si Ow 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.

8. References

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

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

[3] Fox, M. S.
The Intelligent Management System: An Overview.
Technical Report, Robotics Institute, Carnegie-Mellon University, August, 1981.

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

[5] Fox, M. S.
Job Shop Scheduling: An Investigation in Constraint-Directed Reasoning.
PhD thesis, Carnegie-Mellon University, 1983.

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

[7] Smith, S.F.
Exploiting Temporal Knowledge to Organize Constraints.
Technical Report, Robotics Institute, Carnegie-Mellon University, To Appear.

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

[9] Sussman, G, J. and G. L Steele, Jr.
CONSTRAINTS--A Language for Expressing Almost-Hierarchical Descriptions.
Artificial Intelligence 14:1-40, 1980.

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

[11] Wright, J.M. and M.S. Fox.
SRL/1.5 User Manual.
Technical Report (Draft), Robotics Institute, Carnegie-Mellon University, 1982.