Appeared In: Porceedings Sixth Canadian Conference on Artificial Intelligence
Sponsored By: Canadian Society for Computerized Studies of Intelligence
Montreal, Quebec - May, 1986

OBSERVATIONS ON THE ROLE OF CONSTRAINTS IN PROBLEM SOLVING

Mark S. Fox

Intelligent Systems Laboratory
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

1. Introduction

Constraints have a larger role to play in heuristic search than has been demonstrated. It is possible that many of the search architecture design decisions may be deduced given a semantically complete description of the problem's constraints. The intent of the research described in this paper is to develop a semantics for the description of constraints, and a search algorithm which uses these constraints to efficiently search the combinatorial solution space.

Le concept de contrainte a un role plus important à jouer dans la recherche à base d'heuristique qu' il ne fut demontré pour l'instant. La conception d'une architecture de recherche peut souvent être déduite d'une description sémantique complète des contraintes du domaine d'application. Le but de la recherche décrite dans cet article est de développer une sémantique de description des contraintes, et un algorithme de recherche qui utilise ces contraintes pour parcourir efficacement l'espace combinatoire des solutions.

1.1. Problem-Solving Architectures

Simon (1983) has proposed that there are three "rather distinct ways ... for representing and thinking about problem solving tasks." The first views problem solving as a *search* through a model space of nodes (i.e., states) and links. The second views problem solving as *reasoning*, where new statements are deduced from a set of axioms in a formal language of logic. The third views problem solving as *constraint satisfaction*, where the incremental addition of constraints narrows down a set of objects to a subset which satisfies all the constraints. While these views are not mutually exclusive, they are viewed as being distinct. In fact, a constraint satisfying algorithm is viewed as not creating new objects, but reducing the entire space of objects to a satisficing set[1] On the other hand, search techniques, such as those used for planning, can be synthetic; incrementally constructing a solution as part of the search process.

Search, coupled with heuristics, has been the most successful of the techniques for solving real, combinatorially complex problems. Examples include:

- Hearsay II: Speech Understanding (Erman et al., 1980)
- Molgen: Molecular experiment planning (Stefik, 1981a)
- ISIS: Job-shop scheduling (Fox, 1983)
- Aladin: Aluminum alloy design (Farinacci et al., 1986)
- R1/XCON: Computer configuration (McDermott, 1983)
- GARI: Process planning (Descotte & Latombe, 1985)

Surprisingly, each of these systems use constraints, in one for another, to guide the search process. Consequently, conti based problem solving is less distinct than one would assu The question is: What role do constraints actually play in sea based problem solving?

1.2. Problems with Heuristic Search

The design of search architectures is an art. Skilled engineers approach a problem with a set of techniques, which example, have been shown useful in other tasks. Th techniques have been acquired through experience. In fact, training of AI engineers is similar to the medival guild syst where apprentices work with masters for some period of ti Only recently, have papers been prepared which bear a simila to a guild member's handbook (Stefik et al., 1981; Kline & Doli 1985). My recent investigations (Fox, 1983; Baykan & Fox, 19 in the use of constraints as the primary representational paradi in solving problems using heuristic search has led me to conclu that constraints have a larger role to play than previou imagined. Specifically, a number of the architectural des decisions can be based upon the knowledge embedded i sufficiently rich constraint representation. This may lead to automation of parts of the problem solving architecture des process[2]

In the rest of this paper, I review the evolution of heuri search and contraint based problem solving techniques. This followed by a historical review of the ISIS job shop schedul system, whose exploration led to many of the ideas formula here. Lastly, observations about constraints are explored w respect to their relevance in determining the architecture o domain specifice problem solver.

2. The Evolution of Heuristic and Constraint-Based Search

If constraints are to play a role in determining the structu of search, it is necessary to identify those structures. This secti reviews the evolving set of heuristic search structures, followed a review of the use of constraints in search.

[1] This assumes the ability to enumerate a set of objects from which to choose.

[2] At this time, I am not so bold to conclude that all of it could be automated.

2.1. Heuristic Search

Search explores a network of states in which each state represents a step along the path to a solution. The most facile use of search *anchors* it at the initial state and generates a tree in a breadth or depth first manner.

Game playing extends the concept of search to include the heuristic rating of states using domain specific knowledge. Due to the large size of the search space, game playing systems are required to prune the examined states. To achieve this, an *evaluation function* is used to rate states, in effect answering: Of all the legal moves that can be made, what are the ***preferred*** moves? A variety of search algorithms have been used such as min-max, A* (Nilsson, 1971), and B* (Berliner, 1979). An evaluation function can be used to measure structure, i.e., ply and fanout, reducing the technique to a breadth or depth first search.

The search techniques described above assume the application of all operators at each state. Means-ends analysis (Newell & Simon, 1956), provides for the selection of the *best operator* to reduce the difference between the current state and the goal state. Consequently, operators are ordered in addition to states.

Early robot planning research resulted in the formalization of operators in the predicate calculus. The STRIPS system (Fikes & Nilsson, 1971) represented operators as rules with *pre-conditions and post conditions*. GPS-like means-end analysis was used to plan tasks.

Simon (1962) recognized that a planning system in a real domain will have to struggle with the size of the search space. He proposed that search be done at differing *levels of abstraction*. By designating search hierarchies, search can proceed at the highest, least detailed level and use the results to constrain search at the next, more detailed level, and so on. One could view the ordering of differences and operators in GPS's difference table as an implicit hierarchy. The first use of this concept was in ABSTRIPS (Sacerdoti, 1974). By separating pre-condition variables into levels of importance, the pre-conditions would contain only the variables important at the current level of planning.

Another type of reasoning with differing levels of abstraction can be found in the Hearsay-II speech understanding system (Erman et al., 1980). These levels were defined by data abstractions.

Goal protection is another issue for search. The result of one action may be reversed by another before the result could be used in the overall achievement of the goal. To deal with this, the HACKER system (Sussman, 1975) used a *debugging* approach to fix a plan after it was constructed. A set of *critics* were dynamically constructed to recognize errors and suggest corrections. The NOAH system (Sacerdoti, 1975) took a *least-commitment* approach to planning. NOAH would not sequence operations unless forced. This approach reduced the amount of backtracking necessary to secure a legal plan because the current plan did not make any unnecessary sequencing decisions.

Hayes-Roth & Hayes-Roth (1980) call the combination of bidirectional problem-solving and the ability to start problem-solving at any point in the search space (island-driving as opposed to left to right) found in Hearsay-II, *opportunistic* reasoning. Opportunistic reasoning reduces the search space by focussing the planning effort in areas that are of high certainty and/or highly constrained. By extrapolating from these "islands", further constraints on the more uncertain parts of the planning space should be generated.

Problem-solving architectures were extended by incorporating the *interacting experts* paradigm. This paradigm first appeared in Hearsay-II in the form of a blackboard architecture in which the experts communicate by generating and testing hypotheses on a shared blackboard. A somewhat different idea appears in Beings (Lenat, 1975) and Actors (Hewitt, 1973) where the experts communicate directly.

Hearsay-II also introduced the concept of *focus of attention* (Hayes-Roth & Lesser, 1976). Policy modules in Hearsay-II determined the sequence of knowledge source executions. When parts of the utterance remained uninterpreted, Hearsay-II dynamically determined what parts of the search space required more attention and turned the systems resources towards reducing the uncertainty in those areas. By understanding what problem-solving methods Hearsay-II had available (i.e., knowledge sources) and its resource constraints, it would decide the best next action. The ability to reason about "how to reason" (or plan) has been called *meta-planning* in MOLGEN (Stefik, 1981b) and also appeared as *meta-rules* in TEIRESIAS (Davis, 1976; Davis & Buchanan, 1977). The implementation of meta-planning in MOLGEN used the concept of levels of representation for operators, in addition to what was commonly found for variables.

The blackboard architecture has been extended to include *multiple blackboards*, some of which are concerned with control, others which are concerned with the problem domain (Hayes-Roth & Hayes-Roth, 1980; Rychener et al., 1986).

Although much of the above search research was concerned with how to reduce the search space, other aspects of the search problem must be considered. Game playing systems introduced search techniques for adversary-oriented games. That is, the search would consider both the programs' moves and the opponents moves to determinine a next move. The concept of *adversary-oriented planning* has reappeared as *counter-planning* in the POLITICS system (Carbonell, 1979). This research can be viewed as a form of goal-protection in which the system has to consider what the adversary may do to prevent the system from achieving its goals.

All of the above search research is concerned with achieving a single goal. But another type of search is concerned with the satisfaction of *multiple, possibly competing goals*. NUDGE is an early system which focused on multiple goal satisfaction (Goldstein & Robert, 1977). A heuristic approach was developed for the domain of appointment calendar maintenance. This research was unique because it included rules for the *relaxation* of constraints. When a schedule could not be found that satisfied the existing constraints, it used the rules to propose alternatives (possibilities) by relaxing certain constraints, such as preferences. In this case, the preference constraint was simply removed. Other rules peculiar to the appointment domain were used to alter existing calendar requirements until a viable schedule was produced.

Aladin, an alloy design system (Rychener et al, 1986) deals with multiple goal protection through *over satisfaction*. By over satisfying a continuous goal initially, other goals which may reduce its satisfaction will not reduce it enough to be "broken".

2.2. Search with Constraints

In parallel, a somewhat divergent set of work has led to an understanding of how to solve problems using constraints. Linear programming , at one end of the spectrum, appears to bear little relation to the AI theory of problem-solving. At the other end is the constraint-directed heuristic search of REF-ARF (Fikes, 1971) and MOLGEN (Stefik, 1981) which combines a constraint representation with heuristic search.

One of the earlier works in constraint analysis was REF-ARF (Fikes, 1970). Its task was similar to the linear programming task. Given a set of linear inequalities that restrict the possible values of a set of variables, can value assignments be found for them? Rather than a brute force search for a set of bindings that satisfied all the constraints (equations), REF-ARF used the constraints to reduce the generated binding set. Hence, the system can be viewed as a classical generate and test, by which the system was able to take the constraints and use them in the generator to reduce the size of the search space.

Another form of constraint is an adjacency network such as a grammar. A grammar defines the legal sentences that can be formed from a symbol set. The grammar can be viewed as a constraint on the symbols that will be recognized and/or generated. It defines what symbols are compatible with other symbols when linearly ordered. Another example is the conceptual hierarchy of the SEMANT knowledge source of Hearsay-II (Fox & Mostow, 1977). It is similar to a grammar, but relaxes the sequence constraint at the phrase level, allowing ungrammatical sentences, and sentence fragments to be understood. A third instance is the 3D space description network used in ARGOS (Rubin, 1978). In this case, a network was used to define adjacencies of objects in a visual scene, and used to constrain the set of acceptable labelings of an image.

In many real-world applications, constraints are not binary, but are continuous. For example, in image understanding, how a pixel is to be labeled is determined by the labels of neighboring pixels. The knowledge of how to do neighborhood based labelling is at best uncertain, hence the constraints that tie pixels together return a certainty rating for each of the possible labellings of the pixel. The higher the rating, the more probable that the label is correct. This type of constraint is the chief mechanism of *relaxation* (Zucker, 1976). Relaxation can also be viewed as a network constraint system. The goal is to assign a value to each node. A nodes value is constrained by the compatibility rules on the incident arcs. The CONSTRAINTS system (Sussman & Steele, 1981) can be viewed (loosely) as the dual of relaxation. Behavior rules are associated with nodes, and values with arcs. When an arc value changes, a node's rules determine its effect (i.e., value) on the other incident arcs. The system could recognize inconsistencies in arc values due to the lack of uncertainty in rule knowledge.

As search moved from single level to hierarchical, so has relaxation and relaxation-like processes. Single level relaxation often did not have enough information to adequately label a scene. By creating multiple levels of representation, higher levels of knowledge could be incorporated (Zucker, 1977).

The next step was to combine both binary and continuous constraints in a hierarchical system. Again, image understanding research has been the area for this research (Ballard et al., 1977; Russell, 1979). The representation of constraints in image understanding has also been extended to predicate calculus. Davis (1980) makes the case that predicate calculus is a better representation for discrete relaxation constraints.

MOLGEN combined planning with constraint-analysis (Stefik, 1981a). As plans were broken into sub-problems, variable value constraints determined in one subproblem were propagated to other subproblems. Hence, variables accumulated constraints across subproblems before an actual binding was chosen (a least commitment approach).

Engelman et al. (1980) in interactive frame instantiation associate constraints with groups of slots. An interesting feature of their approach is that constraints form buckets, each having its own priority. Hence, constraints have a priority ordering.

In planning driving paths through a town, McCalla (1978) considered constraints such as possible routes, and time and space restrictions.

Fukumori (1980) used a constraint-based approach to determine the arrival and departure times of trains at stations. Trains initially have fuzzy times assigned (i.e., a time span or *belt*). Constraints then reduce the size of the belt. The problem is much simpler than the general scheduling problem: trains had only one route and two resources, a track and stations. The fuzziness of times is similar to that used in Hearsay-II to denote the time span of an hypothesis when its boundaries were uncertain.

The GARI system (Descotte & Latombe, 1985) combines constraint satisfaction with least committment. By evaluating constraints in priority order, precendence was introduced into parallel process plans. If the problem became overconstrained, the last constraint introduced, which is the lowest priority constraint, would be relaxed.

The WRIGHT system (Baykan & Fox, 1986) approaches the problem of kitchen design as a search problem where constraints define the search operators. Each constraint has a measure of uncertainty which signifies the level of uncertainty in the search space if the constraint is satisfied. Hence, the most certain constraint is chosen to satisfy at each step, with the hope the the resultant search space will be less complex.

2.3. Topological Assumption

The principal assumption which underlies, though implicitly, the success of these search structures is that *understanding a problem's search space will enable the selection of effective and efficient search structures.* Constraints appear to participate directly in search, as evaluation functions, as operators, and in other important ways, which will be discussed in the rest of the paper.

AI shares this assumption with Operations Research (OR). An examination of mathematical programming recognizes that OR has also been pursuing the problem of how to satisfy constraints

in combinatorially complex search spaces. The Simplex and the Bell Labs algorithms for solving linear constraint problems are the result of a topological analysis of the search space. Simplex identifies that an optimal solution can be found by visiting the vertices, while the Bell Labs solution assumes a multi-dimensional topology where jumps can be made through space between points.

OR's success in mathematical programming is restricted to the special class of linear problems. For non-linear problems, OR uses heuristics to guide the search process. In fact, recent advances appear to parallel those of AI. For example, Lagrangian relaxation is a hierarchical search technique which abstracts the complex non-linear problem to a higher level linear model whose solution guides search at the next level.

The constraint perspective investigated in the rest of the paper can be viewed as being in contention with that of Lenat (1982). He believes that the structure of search is unimportant; knowledge is everything. He fails to recognize that implicit in his heuristics is knowledge of structure. Just as Lenat looks for a richness in the representation of heuristics, I am looking for a richness in the representation of constraints, and an understanding of how they impact the structures used in a problem solving architecture.

3. Role of Constraints in Job-Shop Scheduling

In 1980, I was asked to explore the application of AI techniques to a turbine component plant's job-shop scheduling problem. The primary product of the plant was steam turbine blades. A turbine blade is a complex three dimensional object produced by a sequence of forging, milling, grinding and finishing operations to tolerances of a thousandth of an inch. Thousands of different styles of blades are produced in the plant, much of them as replacements in turbines currently in service.

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

1. Forced outages (FO): Orders to replace blades which malfunctioned during operation. It is important to ship these orders as soon as possible, no matter what the cost.

2. 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.

3. Service and shop orders (SO, SH): Orders for new turbines. Lead times of up to three years may be known.

4. Stock orders (ST): Order for blades to be placed in stock for future needs.

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

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

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

In the plant, each part number has one or more process routings containing ten or more operations[3]. Process routing variations may be as simple as substituting a different machine, or as complex as changing the manufacturing process. Further more, the resources needed for an operation may also be needed by other operations in the shop.

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

- It is a *time-based* planning problem (i.e., scheduling) in which activities must be selected, sequenced, and assigned resources and time of execution.

- It is a *multi-agent* planning problem. Each order represents a separate agent for which a plan/schedule is to be created. The number of agents to be scheduled is in the hundreds.

- The agents are *uncooperative*. Each is attempting to maximize its own goals.

- *Resource contention* is high, hence closely coupling decisions.

- Search is *combinatorially explosive*. 85 orders moving through ten operations without alternatives, with a single substitutable machine for each operation and no machine idle time has over 10^{880} possible schedules.

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

From this analysis, I concluded that the object of scheduling is not only meeting due dates, but satisfying the many constraints found in various parts of the plant. Scheduling is not a distinct function, separate from the rest of the plant, but is highly

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

connected to and dependent upon decisions being made elsewhere in the plant. The added complexity imposed by these constraints leads schedulers to produce inefficient schedules. Indicators such as high work-in-process, tardiness, and low machine utilization support this conclusion[4]. Hence, any solution to the job-shop scheduling problem must identify the set of scheduling constraints, and their affect on the scheduling process.

Once the issue of designing a constraint-directed scheduling system was identified, a decision was made to solve the problem by constructing the ISIS family of systems. The purpose of the family is to investigate the performance of successively more sophisticated search architectures (Fox & Reddy, 1981). At each stage experiments have been run to measure the effectiveness of the architecture[5]. The rest of this section describes the family of systems called ISIS.

3.1. ISIS-0

3.1.1. ISIS-0 Goals

In the fall of 1980, work began on ISIS-0. The purpose was to identify the types of constraints used by schedulers, and the extent to which they could prune the space of alternative schedules.

3.1.2. ISIS-0 System Architecture

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

- Each order was scheduled separately, in priority order, as determined by a combination of order category and due date.
- Search could be performed forward from the order's start date or backward from the order's due date.
- Operators would generate alternative operations, machines, and operation times. The shop was loaded hence the availability of resources at a particular time was restricted.
- States represent partial schedules. A path through the network determines a complete schedule.
- Constraints were either imposed exogenously by the scheduling person upon the system, or were already embedded in the factory model and their applicability determined at each point in the search space.
- Propagation of constraints was performed when scheduling decisions early in the search path restricted decisions further on.

ISIS-0 was completed in December 1980 and partially demonstrated, bugs and all, at the sponsoring plant.

3.1.3. Constraint Categories

Research on version 0 of ISIS yielded five broad categories of constraints. The first category encountered is what I 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-Process**: Work-in-process (WIP) inventory levels are another concern. WIP inventory represents a substantial investment in raw materials and added value. These costs are not recoverable until delivery. Hence, reducing WIP time is desirable.
- **Resource Levels**: Another concern is maintaining adequate levels of resources necessary to sustain operations. Resources include personnel, raw materials, tools, etc. Each resource will have associated constraints. For example, labor size must be smoothed over a month's interval, or raw materials inventory may have to be limited to a two day supply.
- **Costs**: Cost reduction can be another important goal. Costs may include material costs, wages, and lost opportunity. Reducing costs may help achieve other goals such as stabilization of the work force.
- **Production Levels**: Advance planning also sets production goals for each cost center in the plant. This serves two functions: it designates the primary facilities of the plant by specifying higher production goals, and also specifies a preliminary budget by predicting how much the plant will produce. One outcome of this activity is a forecast of the work shifts that will be run in various areas of the plant.
- **Shop Stability**: Shop stability is a function of the number of revisions to a schedule and the amount of disturbance in preparation caused by these revisions. It is an artifact of the time taken to communicate change in the plant and the preparation time.

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

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

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

[5] Too few AI systems today attempt to measure their effectiveness.

processing times associated with different manufacturing operations.

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

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

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

A fifth category of constraint is *preference*. 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 lists the variety of constraints we have identified as well as the categories we have used to classify them.

Constraint	Org. Goal	Physical	Causal	Pref.	Avail.
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		

3.1.4. Constraint-Directed Search Concepts

Constraint-Directed Evaluation. ISIS-1 dynamically constructed a different evaluation function for each state in the search space. It constructs the evaluation function out of the constraints which have been resolved to be applicable to the state under consideration. Each constraint contributed both an importance (i.e., weight) and utility. Constraints were resolved by extracting them from the resources and operations defined in the particular state.

3.2. ISIS-1

3.2.1. ISIS-1 Goals

ISIS-0 identified the broad categories of constraints but more work on representation and search architecture was required. In January 1981, work on the second version of ISIS began. The intent of this system was twofold. Given the central role of constraints to determine a job shop schedule, a major thrust of our research focused on the identification and characterization of 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:

- How important is the constraint relative to the other known constraints? Is it more important to satisfy the cost constraint than the due date?
- If I cannot find a schedule which satisfies the constraint, are there relaxations of the constraint which can be satisfied. I.e., is there another due date which is almost as good?
- If there are relaxations available for the constraint, are any more preferred? Perhaps I would rather ship the order early rather than late.
- If I chose a particular relaxation, how will it affect the other constraints I am trying to satisfy? Will meeting the due date negatively or positively affect the cost of the order?
- Under what conditions am I obliged to satisfy a constraint? What if there are two constraints specified for the same variable, i.e., two different due date for the same lot? Or there may two different due dates depending on the time of year.

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

The second goal was to measure the effectiveness of the a modified Beam search (Lowerre, 1976) architecture which uses constraints.

3.2.2. ISIS-1 System Architecture

The salient points of the architecture include:

- Search is divided into three levels: Order selection, resource analysis, and resource assignment.

- Each level is composed of three phases: A pre-search analysis phase which constructs the problem, a search phase which solves the problem, and a post-search analysis phase which determines the acceptability of the solution. In each phase, ISIS-1 uses constraints to bound, guide, and analyze the search.
- The order selection level is responsible for selecting the next unscheduled order to be added to the existing shop schedule. Its selection is made according to a prioritization algorithm that considers order type and requested due dates. The selected order is passed to the resource analysis level for scheduling.

- The resource analysis level selects a particular routing for the order and assigns reservation time bounds to the resources required to produce it. Pre-search analysis begins with an examination of the order's constraints, resulting in the determination of the scheduling direction (either forward from the start date or backward from the due date), the creation of any missing constraints (e.g. due dates, work-in-process), and the selection of the set of search operators which will generate the search space. A beam search is then performed using the selected set of search operators. The search space to be explored is composed of states which represent partial schedules. The application of operators to states results in the creation of new states which further specify the partial schedules under development. Depending on the results of pre-search analysis, the search proceeds either forward or backward through the set of allowable 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 a machine and/or execution time to the operation. 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. 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 resource selection level. This enables the adjustment of reservations in order to reduce work-in-process time. Alternative resources (e.g. tools, materials, etc.) are generated by other operators. Each state in the search space is rated by the set of constraints found (resolved) to be relevant to the state and its ancestors. This set is determined by collecting the constraints attached to each object (e.g. machine, tool, order, etc.) specified by the state and applying resolution mechanisms. Each constraint assigns a utility between 0 and 2 to a state; zero signifies that the state is not admissible, 1 signifies indifference, 2 maximal support. The rating of a state with multiple constraints is the mean of the utilities assigned by the constituent constraints, each weighted by the the importance of the assigning constraint.

  Once a set of candidate schedules has been generated, a rule-based post search analysis examines the candidates to determine if one is acceptable (a function of the ratings assigned to the schedules during the search). If no acceptable schedules are found, then diagnosis is performed. First, the schedules are examined to determine a type of scheduling error and the appropriate repair. Intra-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 rescheduling 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.

  This level outputs reservation time bounds for each resource required for the operations in the chosen schedule.

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

- In addition to incrementally scheduling orders for production as they are received by the shop, the ISIS-1 search architecture could be exploited in a reactive manner. As unexpected events (e.g. machine breakdowns) cause disruptions in the existing shop schedule, ISIS-1 needed only to reschedule the affected orders. Previous reservations were transformed into preference constraints so that the new search for a schedule for the affected order would follow as much as possible to original schedule. This results in a minimal amount of change, and provides continuity in the shop schedules generated over time.

3.2.3. Constraint Representation

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 Fox (1983) and Smith (1983)).

One of the central issues that must be addressed by the constraint representation is *conflict*. Consider cost and due-date constraints. The former may require reduction of costs while the latter may require shipping the order in a short period of time. To accomplish the latter, faster, more expensive machines may be required, thereby causing a conflict with the former. In short, it may not be possible to satisfy both constraints, in which case one or both must be relaxed. This is implicitly accomplished in

mathematical programming and decision theory by means of utility functions and the specifications of relaxation through bounds on a variable's value. In AI, bounds on a variable are typically specified by predicates (Engleman, 80; Stefik, 81) or choice sets (Steele, 80; Waltz, 75).

Given the diverse 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 utility of a relaxation may have more than one interpretation, which can be problematic. In the case of the due date constraint, it represents a preference for shipping on time rather than late. In the case of shifts it represents the degree of difficulty with which another should can be added. In both cases, the focus is on the difference in utility between alternative relaxations. This difference is called the *elasticity* of the relaxation. The greater the decrease in utility, the lower the elasticity. If the information were available, the utility measure would reduce to a cost function.

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 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.

Consider a constraint that restricts the length of a turbine blade that can be milled on a machine to less than 28.5 inches. This can be represented by the schema **product-length-requirement** which is a combination of a **required-constraint** and a **binary-attribute-constraint.**

product-length-requirement specifies that the foil-length of a blade is being constrained. It is obligated to being used during an airfoil-operation, and it negatively affects the airfoil-machine-preference constraint. The actual constraint is specified in **product-length-constraint.** If the constraint is satisfied then the utility returned will be 1.2, otherwise 0. The predicate of the requirement is specified by the **product-length-predicate** It specifies that any blade must have a foil-length less than 28.5 units. One potential problem in constructing this constraint is enabling the predicate to refer to slots in the root constraint (i.e., product-length-requirement). You will notice that the predicate schema is linked to the requirement schema via a PREDICATE-OF relation (inverse of predicate), and that the requirement schema is linked to the range constraint by a CONSTRAINT-OF relation (inverse of constrained-by). Each of these relations allow the inheritance of slots and values. Hence the **product-length-predicate** inherits the DOMAIN, and RELATION slots from **product-length-requirement.**

The **product-length-requirement** is not attached to the FOIL-LENGTH slot of all products, but is attached instead to the

airfoil-operation schema (i.e., contained in the constraint slot). It is up to ISIS to retrieve the constraint from the operation's CONSTRAINT slot, and apply its tester function to the search state

```
{{ product-length-requirement
    IS-A: range-constraint
    DURING: airfoil-operation
    CONSTRAINS: airfoil-machine-preference
        direction: neg
    DOMAIN:
        range: (type "is-a" "blade")
    RELATION: foil-length
    CONSTRAINED-BY: product-length-constraint }}
```

Figure 3-1: product-length-requirement Schema

```
{{ product-length-constraint
    CONSTRAINT-OF: product-length-requirement
    INSTANCE: required-constraint
    RELAXATION-TYPE: required
    TRUE-UTILITY: 1.2
    PREDICATE: product-length-predicate }}
```

Figure 3-2: product-length-constraint Schema

```
{{ product-length-predicate
    PREDICATE-OF: product-length-constraint
    INSTANCE: binary-attribute-predicate
    RANGE-2: 28.5
    PREDICATE: lessp }}
```

Figure 3-3: product-length-predicate Schema

and constraint. The tester retrieves the blade being scheduled and places it in the domain slot, and applies the contents of the APPLY slot in the predicate to its schema.

Another manufacturing constraint is the specification of shifts. A shift defines the time that a work center is availabe for work. Historically, it has been discrete, specifying one, two, or three shifts during a work day. In addition, the number of shifts on a week end may differ from that during a week day. Therefore, a shift constraint should specify what the normal available shifts are, what the relaxations are, and the period during which the shift constraint should be interpreted.

A shift specification may be specified as a **discrete-constraint**. The CONSISTENCY of the slot is exclusive, specifying that only one shift constraint may exist for the slot. No alternatives are specified at this point.

```
{{ shift-constraint
    IS-A: range-constraint
    DOMAIN:
        range: (or (TYPE is-a machine) (TYPE is-a work-center))
    RELATION: shift }}
```

Figure 3-4: shift-constraint Schema

```
{{ shift-constraint-spec
    is-a: discrete-constraint
    CONSISTENCY: exclusive }}
```

Figure 3-5: shift-constraint-spec Schema

```
{{ shift
    START-TIME:
    END-TIME:
    WORK-WEEK: }}
```

Figure 3-6: shift Schema

An example of a shift constraint is that specified for a wmf1 machine

```
{{ wmf1-shift
    IS-A: shift-constraint
    DOMAIN: wmf1
    RELATION: shift
    CONSTRAINED-BY: wmf1-shift-constraint }}
```

Figure 3-7: wmf1-shift Schema

The range constraint specifies the domain of the constraint and relation. That is, the constraint affects the SHIFT slot of the **wmf1**. The contents of the CONSTRAINED-BY slot is the name of the constraint: **wmf1-shift-constraint**. It describes a start-time, end-time, and day for the shift.

The shift constraint is not a schema constraint. Each relaxation completely specifies the start-time, end-time, and work week. They cannot be relaxed individually. The contents of the RELAXATION slot specify another shift, **wmf1-shift-relaxation**, to be used in addition to the first constraint (the DISCRETE-TYPE of the constraint is inclusive).

The constraint is interpreted by taking the value of the TESTER slot from **wmf1-shift** (not shown) and applying it to the pair (<state> **wmf1-shift**). The tester will retrieve the discrete constraint and find the value which matches the value under

```
{{ wmf1-shift-constraint
    INSTANCE: shift-constraint-spec
    RELAXATION-VALUE: {{ INSTANCE shift
                         START-TIME:    8:00
                         END-TIME:      16:00
                         WORK-WEEK:     (OR monday tuesday wed
                                          thursday friday)   }}
    RELAXATION-UTILITY:   2
    RELAXATION: wmf1-shift-relaxation }}
```

Figure 3-8: wmf1-shift-constraint Schema

consideration (i.e., specified inthe state) and return the relaxation utility.

```
{{ wmf1-shift-relaxation
    INSTANCE: shift-constraint
    RELAXATION-VALUE: {{ INSTANCE shift
                         START-TIME:     16:00
                         END-TIME:       24:00
                         WORK-WEEK:      (OR monday tuesday wed
                                             thursday friday)    }}
    RELAXATION-UTILITY:     1.2
    DISCRETE-TYPE:          inclusive    }}
```

Figure 3-9: wmf1-shift-relaxation Schema

The basic due-date-constraint is a continuous value constraint which constrains the due-date slot of a lot. The choice of a due-date has a utility specified by the PIECE-WISE-LINEAR-UTILITY. The utility is specified by (shipping-lateness utility) pairs. An example of its use is a due date for forced outage orders. The tester for **due-date-constraints** takes the search state and the constraint as parameters, retrieves the due date being considered int the state, or predicts one, and applies the value of the utility function slot to the due date. The utility function uses the PIECE-WISE-LINER-UTILITY value to interpolate and return a utility.

fo-due-date specifies that the utility of the due date chosen is 2 if it less than or equal to the the requested due date. It is linearly decreasing to 0.2 if it greater than 0 days late and less than 7. And is 0.2 if greater than 7 days late.

```
{{ due-date-constraint
    IS-A: range-constraint
    DOMAIN:
        range: (type "is-a" "lot")
    RELATION: due-date
    CONSTRAINED-BY:
        range: (type "is-a" "due-date-constraint")
    TESTER: due-date-tester
    PRIORITY-CLASS:  }}
```

Figure 3-10: due-date-constraint Schema

```
{{ due-date-constraint-spec
    IS-A: continuous-constraint
    CONSISTENCY: exclusive
    UTILITY-FUNCTION: interpolate
    PIECE-WISE-LINEAR-UTILITY: }}
```

Figure 3-11: due-date-constraint-spec Schema

```
{{ fo-due-date
    IS-A: due-date-constraint
    PRIORITY-CLASS: forced-outage
    CONSTRAINED-BY: {{ INSTANCE due-date-constraint
                       PIECE-WISE-LINEAR-UTILITY: ( (0 2) (7 0.2)
    }}
```

3.2.4. Performance of ISIS-1

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

A number of experiments were performed. These experiments explored the effects of alternative constraints, alternative search operators, and beam width size. A detailed discussion of all experiments may be found in Fox (1983).

The gantt chart[6] shown in **Figure 3-1** depicts a schedule generated by ISIS-1.

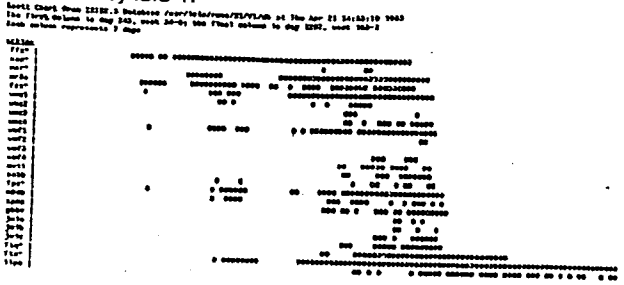

Figure 3-1: Version 1 Gantt Chart

The schedule is a poor one; 65 of the 85 orders scheduled were tardy. To compound the problem, order tardiness led to high work-in-process times (an average of 305.15 days) with an overall makespan[7] 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 3-1) 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.

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

3.2.5. Constraint-Directed Search Concepts

Constraints as Generators. Constraints which specify precedence between operations and requirements for resources can be interpreted as search operators. For example, a constraint which specifies that drilling must following milling can be interpreted as operator which extends a state for which milling is defined to be the operation into a new state for which drilling is the successor operation. Each constraint in ISIS-1 has code which interprets the constraint as an operator to be used in search.

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

[7] Makespan is the time taken to complete all orders.

ISIS-1's presearch analysis selects the operators from a subset of available constraints.

Constraints Bound the Search Space. The omission of constraints by pre-search analysis (e.g., alternative shifts), when defining operators, results in a bounding of the search space. This restriction on the size of the search space is intentional but can be relaxed by post-search analysis.

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

Constraint Resolution and Dynamic Evaluation. ISIS-1 extends the concept of dynamic evaluation function construction by utilizing a more sophisticated form of constraint resolution.

Local Resolution. ISIS-2 dynamically resolves the set of applicable constraints at each search state. Resolution is performed by examining each schema (i.e., operation, machine, etc.) in the current state description. The contents of any CONSTRAINT slots, or constraints attached to any slots which enable the schema are added to the local resolution set. Constraints may originate from four sources:

Model-Based: Constraints may be embedded in any resource or activity in the factory model. For example, there may be physical constraints associated with a machine, sequencing constraints associated with an operation, queue ordering constraints associated with certain work centers.

Lateral Imposition: Constraints can also be propagated laterally during the search. A decision made earlier in the elaboration of a schedule may result in a constraint being attached to the lot that restricts a choice point further on in the search.

Exogenous Imposition: The user may also create and implant constraints. These constraints can be attached to anywhere in the model, or be globally attached so that it is considered at each search state.

Global Resolution. The rating of a state is a rating of the partial schedule up to the current state, and not the single choice represented by the state. Hence, the rating of a state must include not only the local constraints but the constraints applied to all the states along the partial schedule ending at the current state. Not all the constraints locally resovled at each state along the path are globally resolved. Consider the **due-date-constraint** (figure 5-10). It is a classic evaluation function as defined in heuristic search. Part of the constraint calculates the work-in-process time of the lot to the current state, and the other part *predicts* the remaining work-in-process time to the end state. Each time the constraint is applied, it is a better estimator of the work-in-process time, and should overide applications of the same constraint earlier in the partial schedule. On the other hand, the **queue-stability** constraint is applied at each state which binds a queue position. It rates the state by how much it destabilizes existing queue reservations. The greater the destabilization, the lower the rating. This constraint measures a decision made at that state, and remains invariant over future states, since any future states cannot affect an earlier state.

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

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

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

3.3. ISIS-2

3.3.1. ISIS-2 Goals

ISIS-1 identified the representational requirements of constraints, and their use in directing search. Neither changes in beam width, nor alterations to existing constraints were able to significantly affect the degree to which due date and work in process constraints were unsatisfied. The cause of this problem lay with the combinatorics of the search space combined with the horizon effect. ISIS-2 was designed reduce the impact the horizon effect has on the quality of the schedules.

3.3.2. ISIS-2 Architecture

ISIS-2 constructs schedules by performing a hierarchical, constraint-directed search in the space of alternative schedules. An additional level was added between order selection and

resource analysis: capacity analysis. The purpose of this level was to consider a subset of the more important constraints in order to "look ahead" so that capacity bottlenecks could be identified in a smaller search space.

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

3.3.3. ISIS-2 Performance

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

The average utility assigned by the due date constraint to lower priority "service orders", for example, almost doubled, rising from a value of 0.46 in the first experiment to a value of 0.80. The total number of tardy orders was reduced to 14. Moreover, a much lower average work-in-process time of 186.73 days was

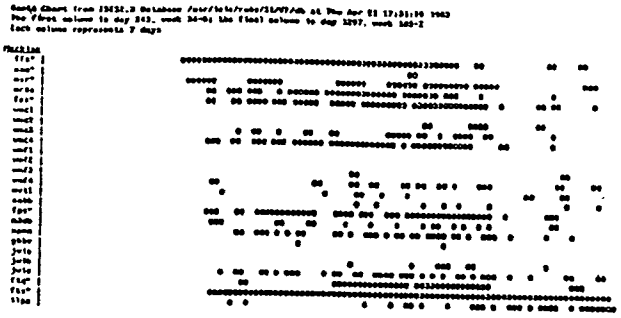

Figure 3-2: Version 7 Gantt Chart

achieved, resulting in an overall makespan of 583.25 days. In this case, inadequate machine capacity in the "final straightening area" (fts*) appeared to be the principal limitation affecting order tardiness.

3.3.4. Constraint-Directed Search Concepts

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

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

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

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

3.4. ISIS-3/OPIS-0

Work began on ISIS-3 (aka OPIS-0) during the summer of 1984. Though ISIS-2 made significant headway in satisfying its constraints in the presence of a high degree of resource contention, it was still believed that better use of the resources could be made resulting in higher constraint satisfaction.

3.4.1. ISIS-3 Goals

The goal of ISIS-3 was to explore the problem of varying perspectives on scheduling (Smith & Ow, 1985). In particular, the high degree of resource contention in multi-agent planning/scheduling forces one to consider scheduling the activities of the resource (i.e., machine) as opposed to the agent (i.e., order). This differs from approaches to problems for which the number of agents are small and resource contention low (e.g., (Konolige & Nilsson, 1980)).

3.4.2. ISIS-3 Architecture

The approach was to mix order scheduling with resource scheduling. (See Smith & Ow (1985) for more details.) This was accomplished as follows:

- The order selection and capacity analysis levels were merged into a single level. This new capacity analysis level used a dispatch rule simulation approach to scheduling all of the orders in parallel in the presence of the same subset of constraints associated with this level.

- The schedule at the capacity analysis level was examined for bottlenecks. Each bottleneck was then scheduled (usually one) resulting in a time at which each order which flows through the bottleneck is to be worked on.

- The bottlenecks and the schedules of orders through them were passed down to the resource analysis level. This level was modified to perform "island driving", similar to that found in Hearsay-II (Erman et al., 1980). Each bottleneck was designated an "island", and the highest priority order was selected and scheduled out (forward and backward) from the island using the original beam search with the added

constraints of this level.

- The rest of the ISIS-2 architecture remained the same.

3.4.3. ISIS-3 Performance

New experiments were performed comparing ISIS-2, ISIS-3 and the COVERT dispatch rule (Ow, 1986). In all cases, ISIS-3 outperformed the other two. The following are some significant measures:

System
ISIS-3
ISIS-2
COVERT

3.4.4. Constraint-Directed Reasoning Concepts

Islands of Certainty: Constraint-directed Focus of Attention. The bottleneck schedule produced by capacity analysis is actually a set of constraints on the search to be performed at the resource analysis level (i.e., each reservation for the bottleneck by an order is an availability constraint). Search at the resource analysis level can identify islands of certainty by the importance and utililty of the constraints. Consequently, by working on each order in priority order, the resource analysis level is able to identify, for that order, the islands of certainty in its search space (i.e., bottleneck reservations) and perform the beam search outward from those islands, resulting in "island driving".

3.5. OPIS-1

The ISIS-3 architecture is still hardwired in the sense that it performs a resource centered analysis at the capacity and level and then an order centered anlaysis at the resource analysis level. Depending on the state of the factory, one of the perspectives may be unnecessary. In the summer of 1985 work began on OPIS, the beginning of a new series of planning/scheduling systems in which opportunism in search places a greater role. In particular, the first version of OPIS, focuses on opportunistic selection of the scheduling perspective. Its architecture bears many similarities to Hearsay-II.

4. Summary

This section has provided an evolutionary view of the ISIS family of job-shop scheduling systems. Two significant results appear during this evolution. First, the development of a semantics for the representation of constraint knowledge, focusing on what is constrained, relaxation, utility, elasticity, importance, interactions and relevance. Secondly, the novel use of constraints in state generation, search space bounding, generative relaxation, resolution, analytic relaxation, periscoping, focus of attention, and stratefication.

5. Role of Constraints in Problem Solving

This section returns to the original hypothesis, that the development of an adequate semantics of constraints will lead to a better understanding of how to define the structure of search. This section ties the semantics of constraints developed in the ISIS family to the search structures of section two. In particular, a number of observations of how the semantics of constraints relates to search structures are explained.

The first five observations define the basic search architecture of states and operators.

Observation 1: "Constraints define the parameters of states in the search space."

Constraints provide a state-space view of problem solving in that they define the variables to be bound by the search process. Examples include:

- due date constrains the date shipped
- shifts constraint constrains the shifts available
- next operation constraints the current operation
- keep cost under x constrains cost of manufacturing

Observation 2: "Constraints define single state generating operators."

The specification that an attribute or relation should be restricted to a single value enables the construction of generators or search operators which will generate a state with the variable being bound to the single value.

For example, if the constraint specifies that the next operation after milling is drilling then an operator can be generated whose action is the generation of a state which binds the operation to drilling when the preceding operation is milling. These leads into the third observation:

Observation 3: "Constraints define the situation or condition of an operator."

Knowledge of constraint *relevance* determines the situation in which the constraint is to be applied. It is straightforward how the relevance knowledge could be transformed into an operator's condition. For example, a third shift may only be available during monday through friday. This condition would be encoded as an operator's condition.

Observation 4: "Complex operators are the combination of two or more constraints."

(This is another version of the question of how to moves tests into a generator.) Within ISIS are two alternative operators which choose a time at which an operation is to be performed, once the operation and machine are bound. These operators are complex; juggling concerns such as setup time reduction, not letting the operation be performed too late, order priority, shop stability, etc. The hand crafting of such an operator can be viewed as the combining of two or more constraints:

- Setup sequencing.
- Shop stability.
- Work in process.
- Order priority.

Observation 5: "Constraints define the evaluation function."

The utility associated with each constraint relaxation, coupled with a constraint's importance provides the basis for an

evaluation function. In particular, they define a linear function which is the weighted average of the utilities of all resolved constraints.

The next three observations focus on the definition of levels within a hierarchical search space.

Observation 6: "Levels of representation are defined by constrained variables part-of hierarchies."

Many variables whose values are constrained participate in part-of hierarchies. For example, the milling machine is part of the milling machine work center, and a day is part of a week. These hierarchies define levels of abstraction for each variable. In the case of scheduling, capacity analysis can be performed using machines, work centers, plants, etc., or time decisions can be made by the hour. date, week, etc.

Observation 7: "Levels of search are defined by the importance of a constraint."

The *importance* of a constraint can be used to determine which variables are to be bound first in a manner similar to that of ABSTRIPS.

Observation 8: "Levels of search are defined by the elasticity of a constraint."

Though a constraint can be relaxed, it may be difficult to do so. For example, it may be easier to ship an order two days later (i.e., relax the due date constraint) than it is to put a third shift on over the weekend. The *elasticity* of a constraint defines another stratification of the search space.

Observation 9: "Levels of search are defined by constraint interactions."

The interdependence of constraints define an interaction hierarchy. For example, the number of shifts available indirectly affect due date and work in process constraints, but not vice versa. One stratefication of the search space would have shift decisions being made at a higher level.

The next two constraints deal with issues of focus of attention.

Observation 10: "Constraints focus attention on islands of cerntainty."

Highly important constraints with low elasticity define decisions which have to conform to the constraint. These constraints define islands of certainty in the search space from which search is to be initiated. For example, if a constraint specifies that an order is to be delivered today and it is the most important constraint without any relaxations, then search begins with that order at the last operation being completed today.

Observation 11: "Constraints direct the diagnosis and repair of poor search decisions."

Diagnosis identifies poor search decisions and repair attempts a correction. In this case, the low utility of a constraint signals a problem, and a constraint's *interaction* with another constraint points to a possible peep hole optimization. Using shifts and due date constraints again, a low due date utility could be corrected by altering the shift decision.

6. Conclusion

It has been the intent of this paper to elucidate the embryo of a theory which unifies constraints with heuristic search. The theory suggests that constraints play an important role in search. That they define much of the structure of the system architecture, for which until now only heuristics existed. As of yet, the theory is incomplete; it is composed of 11 observations. Further work awaits in the elaboration of constraint semantics and the development of an interpreter which will solve a problem given a complete constraint set.

Acknowledgements

The work described in this paper represents the contributions of many people over the years. First and foremost, Steve Smith has contributed an enormous amount to the success of the project, and continues to do so as leader of the OPIS project. Portions of section 3 are based joint writings. Other notable contributers include Brad Allen, Bruce McClaren, Tom Morton, Linda Quarrie, Peng Si Ow, and Gary Strohm.

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

References

Ballard D.H., C.M. Brown, and J.A. Feldman, (1977), "An Approach to Knowledge-Directed Image Analysis", *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, Cambridge MA.

Baykan C.A., and M.S. Fox, (1986), "Facility Design: An Investigation into Opportunistic Constraint Satisfaction", Technical Report, Robotics Institute, Carnegie-Mellon University, Pittsburgh PA, in preparation.

Berliner H.J., (1979), "The B* Tree Search Algorithm: A Best First Proof Procedure", *Artificial Intelligence*, Vol. 12.

Carbonell J.G., (1979), "The Counterplanning Process: A Model of Decision Making in Adverse Situations", Technical Report, Computer Science Department, Carnegie-Mellon University, February 1979.

Davis L., (1980), "A Logic Model for Constraint Propagation", Technical Report TR-137, Computer Sciences Dept., University of Texas, Austin Texas.

Davis R., (1976), "Applications of Meta Level Knowledge to the Construction, Maintenance, and Use of Large Knowledge Bases", Ph.D. Thesis, Computer Science Department, Stanford University. Report No. STAN-CS-76-552.

Davis R. and B. Buchanan, (1977), "Meta-level Knowledge: Overview and Applications," *Fifth International Joint Conference on Artificial Intelligence*, Cambridge MA, August 1977.

Descotte Y., and J-C Latombe, (1985), "Making Compromises among Antagonist Constraints in a Planner", *Artificial Intelligence*, Vol 27, pp. 183-217.

Engleman C., E. Scarl, and C. Berg, (1980), "Interactive Frame Instantiation", *Proceedings of the American Association for Artificial Intelligence*, pp. 184-186, Stanford University.

Erman L.D., F. Hayes-Roth, V.R. Lesser, and D.R. Reddy, (1980), "The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty", *Computing Surveys*, Vol. 12, No. 2, June 1980, pp. 213-253.

Farinacci M.L., M.S. Fox, I. Hulthage, M.D. Rychener, (1986), "The Development of ALADIN, An Expert System for Aluminum Alloy Design", *Artificial Intelligence in Manufacturing*, Thomas Bernold, (Ed.), Springer-Verlag, to appear.

Fikes R.E., (1970), "REF-ARF: A System for Solving Problems Stated as Procedures", *Artificial Intelligence*, Vol. 1, pp. 27-120.

Fikes R.E., and N.J. Nilsson, (1971), "Strips: A New Approach to the Application of Theorem Proving to Problem Solving," *Artificial Intelligence*, Vol. 2,pp 189-208.

Fox M.S., (1983), "Constraint-Directed Search: A Case Study of Job-Shop Scheduling", (Ph.D. Thesis), Technical Report, Computer Science Dept., Carnegie-Mellon University, Pittsburgh PA.

Fox M.S. and D.J. Mostow, (1977), "Maximal Consistent Interpretations of Errorful Data In Hierarchically Modelled Domains", *Fifth International Joint Conference on Artificial Intelligence*, Cambridge MA, 1977.

Fox M.S., and D.R. Reddy, (1981), "Constraint-Based Scheduling in an Intelligent Logistics Support System: An Artificial Intelligence Approach", Proposal to the Air Force Office of Scientific Research, August 1981.

Fukumori K., (1980), "Fundamental Scheme for Train Scheduling", MIT AI Memo No. 596, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge MA.

Goldstein I.P., and R.B. Robert, (1977), "NUDGE: A Knowledge-Based Scheduling Program", MIT AI Memo 405, Cambridge MA.

Hayes-Roth B., and F. Hayes-Roth, (1979), "A Cognitive Model of Planning", *Cognitive Science*, Vol. 3, No. 4, pp. 275-310.

Hayes-Roth F., and V.R. Lesser, (1976), "Focus of attention in a distributed logic speech understanding system", *Proceedings of the 1976 IEEE International Conference on Acoustics, Speech and Signal Processing*, Philadelphia, 1976, 416-420.

Hewitt C., (1973), The ACTOR Formalism. *Third International Joint Conference on Artificial Intelligence*, Stanford, CA.

Kline P.J., and S.B. Dolins, (1985), "Choosing Architectures for Expert Systems", Technial Report RADC-TR-85-192, Rome Air Development Center, Griffiss AFB, NY. (prepared by Texas Instruments Inc.)

Konolige K., and N.J. Nilsson, (1980), "Multiple-Agent Planning Systems", *Proceedings of the First Annual Conference of the American Association for Artificial Intelligence*, August 1980, pp. 138-141.

Lenat D., (1975),"Beings: Knowledge as Interacting Experts", *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, Tiblisi, USSR.

Lenat D.B., (1982), "The Nature of Heuristics", *Artificial Intelligence*, Vol 19, pp. 189-249.

Lowerre B., (1976), "The HARPY Speech Recognition System", (Ph.D. Thesis), Tech. Rep., Computer Science Dept., Carnegie-Mellon University, Pittsburgh PA.

McCalla G., P. Schneder, R. Cohen, and H. Levesque, (1978), "Investigations into Planning and Executing in an Independent and Continuously Changing Microworld", AI Memo 78-2, Dept. of Computer Science, University of Toronto.

Newell A., and H.A. Simon, (1956), "The Logic Theory Machine: A Complex Information Processing System", *IRE Transactions on Information Theory*, Vol. IT-2, No. 3, pp. 61-79.

Nilsson N.J., (1971), *Problem-Solving Methods in Artificial Intelligence*, New York, N.Y.: McGraw-Hill.

Rubin S., (1978), "The ARGOS Image Understanding System", Ph.D. Thesis, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA.

Russell D.M., (1979), "Where Do I Look Now?: Modelling and Inferring Object Locations by Constraints", Technical Report, Computer Science

Department, University of Rochester, Rochester NY.

Rychener M., M. Farinacci, I. Hulthage, and M.S. Fox, (1986), "Constraint-Directed, Hierarchical Search in Aladin, An Alloy Design System", Technical Report, Robotics Institute, Carnegie-Mellon University, Pittsburgh PA, in preparation.

Sacerdoti, E.D., (1974), "Planning in a Hierarchy of Abstraction Spaces", *Artificial Intelligence*, Vol 5, no.2.

Sacerdoti E.D., (1975), "A Structure for Plans as Behavior", (Ph.D. Thesis), Computer Science Dept., Stanford University.

Simon H.A., (1962), "Scientific Discovery and and the Psychology of Problem Solving", *Minds and Cosmos*, Kolodny (Ed.).

Smith S.F., (1983), "Exploiting Temporal Knowledge to Organize Constraints" Technical Report, Robotics Institute, Carnegie-Mellon University, Pittsburgh PA.

Smith S.F., and P.S. Ow, (1985), "The Use of Multiple Problem Decompositions in Time Constrainted Planning Tasks", *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, 18-23 August 1985, pp. 1013-1015, Los Altos CA: Morgan Kauffman Pub Inc.

Stefik M., (1981a), "Planning with Constraints (MOLGEN: Part 1)", *Artificial Intelligence*, Vol. 16, pp. 111-140.

Stefik M., (1981b), "Planning and Meta-Planning (MOLGEN: Part 2)", *Artificial Intelligence*, Vol. 16, pp. 141-170.

Stefik M., J. Aikins, R. Balzer, J. Benoit, L. Birnbaum, F. Hayes-Roth and E. Sacerdoti, (1982), "The Organization of Expert Systems, A Tutorial", *Artificial Intelligence*, Vol 18, pp. 135-174.

Steele G.L., (1980), "The Definition and Implementation of a Computer Programming Language Based on Constraints", (PhD Thesis), MIT tech. rep. AI-TR-595, Cambridge MA.

Sussman G., (1975), *A Computational Model of Skill Acquisition*, New York: American Elsevier.

Waltz D., (1975), "Understanding Line Drawings of Scenes with Shadows", in P.H. Winston (Ed.), *The Psychology of Computer Vision*, New York N.Y.: McGraw-Hill.

Zucker S.W., (1976), "Relaxation Labelling and the Reduction of Local Ambiguities", In *Pattern Recognition and Artificial Intelligence*, C.H. Chen (Ed.), New York: Academic Press.

Zucker S.W., (1977), "Vertical And Horizontal Processes in Low Level Vision", Report No. 77-4, Electrical Engineering Dept., McGill University, Montreal, Quebec.