Job-Shop Scheduling: An Investigation in Constraint-Directed Reasoning

Mark S. Fox, Brad Allen, Gary Strohm
Intelligent Systems Laboratory
The Robotics Institute
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

1. Introduction

This paper describes ISIS-II*, a constraint-directed reasoning system for the scheduling of factory job-shops. ISIS-II takes a heuristic search approach to generating schedules. The key features of ISIS-II's approach is that it can represent and use a variety of different types of constraints to guide the search, and is able to selectively relax conflicting constraints.

The plant under consideration** represents one of the most complex of scheduling tasks. The plant produces thousands of different parts, some of which are similar, some of which are not. Any part can be ordered in quantities from one to hundreds. Each part number has one or more process routings containing at least ten operations. A process routing may differ simply in machine substitutability, or may represent a totally different manufacturing process. Each operation in a process routing requires resources such as machines, tools, operators, fixtures, materials, etc. At any time there are over 200 orders in the plant, each competing for the same resources.

This scheduling problem has been described as NP-hard. The simple sequencing (without gaps or alternative routings) of 10 orders on 5 machines can result in $(10!)^5$ possible schedules. Rather than do simple capacity analysis, as found in the majority of vendor scheduling systems, or use a local dispatch rule approach as found in operations management research, a constraint-directed reasoning approach was chosen. It was found that schedulers spend 80%-90% of their time determining the constraints in the environment that will affect their scheduling decision, and 10%-20% of their time actually constructing and modifying schedules. Any system that was to adequately schedule such an environment must attend to the multitude and variety of constraints.

*Intelligent Scheduling and Information System, Version 2. This research was supported, in part, by the Westinghouse Corporation, and the affiliates program of the Robotics Institute.

**The Westinghouse Turbine Component Plant in Winston-Salem NC.

The rest of this paper describes how ISIS-II represents, searches with, and relaxes constraints in the process of scheduling a job-shop.

2. Constraint Identification

The first step in the construction of ISIS-II was to determine the categories of constraints a scheduler considers. Four categories were distinguished. The first are organizational goals. They include job tardiness, work in process, resource levels, cost, production levels, and shop stability. One can view these constraints as being approximations of a simple profit constraint. The goal of the 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, further profits. The longer the work in process time, the greater the carrying charge for raw materials and value-added operations. Maintaining a designated production level may amortize the cost of the capital equipment in a uniform manner. In practice, most of these costs cannot be accurately determined, but must be approximated. These approximations have resuled in the incorporation of the above constraints in the plant's operating goals.

Physical constraints are a second category. They specify what an object can or cannot be used for. For example, a milling machine may be limited in the size of turbine blade it can work on due to the length of its workbed. On the other hand, a drill may have a function or a graph that defines how long the drill can be run at a particular speed in a particular material.

Gating constraints are a third category. They define the conditions to be satisfied before an object can be used or a process begun. Examples of gating constraints are operation precedence and resource requirements.

Preference constraints are a fourth category. They provide the means by which preferences can be expressed. Machine preferences are one example, operation and queue position preferences are others. A preference can be viewed as a 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 the supervisor does not have an actual cost or quality constraint

to use due to lack of data.

3. Constraint Representation

A constraint may have one of two effects on a schedule. It may determine the admissability of a schedule, or it may determine the acceptability of a schedule. Admissability determines the legality of a schedule against constraints that cannot be relaxed. Acceptability rates a schedule, allowing alternatives to be distinguished. In a domain with many constraints of a possibly conflicting nature, a constraint must specify three things: what it is constraining, what the alternatives of the constraint are if it cannot be satisfied, and how well the object of the constraint satisfies them.

The general constraint schema contains three slots (figure 3-1). The PRECONDITION defines the applicability of a constraint. The evaluation-function, when evaluated, provides a rating of the decision. The WEIGHT denotes the relative importance of the constraint.

```
{{ constraint
    PRECONDITION:
    EVALUATION-FUNCTION:
    WEIGHT: }}
```

Figure 3-1: constraint Schema

How well a decision satisfies a constraint is represented by the rating. A constraint may return a rating in the interval (0,2). 0 denotes rejection, 1 denotes indifference, and 2 denotes maximal support.

More than one type of constraint relaxation is distinguished. A constraint can be Binary, or offer two or more Choices. Binary constraints represent either a Preference or a Requirement. The former rates (acceptability), the latter prunes (admissability). An attribute-restriction schema is a Requirement. It defines a test on an attribute. An example is the length constraint test for a machine (figure 3-2). It defines the blade is to have a foil-length of less than 28.5 (inches).

```
{{ length-constraint
    { INSTANCE attribute-restriction
        OBJECT:           blade
        ATTRIBUTE:        foil-length
        ATTRIBUTE-VALUE:  28.5
        PREDICATE:        lessp } }}
```

Figure 3-2: length-constraint Schema

A choice-constraint specifies a constraint, its relaxations and their utilities. Choices can be discrete or continuous.

```
{{ discrete-constraint
    { IS-A choice-constraint
        ALTERNATIVE:
            Restriction: (TYPE INSTANCE choice-constraint)
        TYPE:
            Restriction: (OR exclusive inclusive)
            Default: exclusive } }}
```

Figure 3-3: discrete-constraint Schema

A discrete-constraint contains an ALTERNATIVE slot which specifies alternative discrete values and their utilities. The TYPE slot defines whether the alternatives are to be exclusive or inclusive. An example of a discrete-constraint is the specification of the number of shifts associated with a particular machine. A shift is represented by a shift schema (figure 3-4). It is a discrete-constraint with the value elaborated into START-TIME, END-TIME, and DAY.

```
{{ shift
    { IS-A discrete-constraint
        (ELABORATE VALUE --> START-TIME END-TIME DAY) }

    MACHINE: }}
```

Figure 3-4: shift Schema

An example of a shift is that specific for a wmf1 machine (figure 3-5).

```
{{ wmf1-shift
    { INSTANCE shift
        MACHINE:    wmf1
        START-TIME: 8:00
        END-TIME:   16:00
        DAY:        (OR monday tuesday wednesday thursday
                        friday)
UTILITY:     2
ALTERNATIVE: {{ INSTANCE shift
                    START-TIME: 16:00
                    END-TIME:   24:00
                    DAY:        (OR monday tuesday wednesday
                                    thursday friday)
                    UTILITY:    1.2
                    TYPE:       inclusive }}
    } }}
```

Figure 3-5: wmf1-shift Schema

An example of a continuous constraint is an order due-date. It specifies the utility of meeting the due date, and the utility of all possible early and late ship dates.

4. Constraint Relaxation

The representation of constraint provides information on how to relax a constraint, and the utility of the relaxation. The decision as to when to relax a constraint is made in two ways.

Generative Relaxation. Constraints are relaxed in a generative fashion during heuristic search. The search operators generate states which represent alternative relaxations of one or more constraints.

Analytic Relaxation. A rule-based system analyzes an order before it is to be scheduled 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.

5. Constraint-Directed Search

ISIS-II constructs a schedule for an order by performing a beam search (Lowerre & Reddy, 1976) in the space of partial schedules. It first performs a pre-search analysis to generate the boundaries of the search space. Followed by a constraint-directed search. And lastly it performs a post search analysis to determine whether the search was effective.

5-1. Defining the Search Space

The first step taken in scheduling an order is to define the problem. That is, to define:

-the constraints that bound the search space, which in turn define the search operators,

-any new constraints that do not already exist,

-the constraint classes to be ignored, and

-a prioritization of the remaining constraints classes.

A rule-based approach is used to examine an order to determine the above. It used information such as order priority, system goals, and previous scheduling history.

An example is the order mfg-order-xxxx (figure 5-1). The order is a "forced outage" which implies that the due-date is a major constraint. The system creates due-date and work in process constraints. It also bounds the search space by only considering alternative operations, machines, and queue positions.

5-2. Searching the Search Space

Before performing the search, a second set of rules examine the order to determine the direction of search (forward from start date, backward from due date, or at some point in a partial schedule), and the initial states in the search. These rules specified the SCHEDULING-DIRECTION and the initial-search-state in figure 5-1. The direction was chosen based on the order's PRIORITY-CLASS.

```
{{ mfg-order-xxxx
   { IS-A manufacturing-order
      PRIORITY-CLASS: forced-outage
      PRIORITY: 7
      STYLE: #7J8924
      ROWS: 12
      DUE-DATE: {{ INTANCE due-date-constraint
                   DATE: (450 0 0) }}
      SEARCH-OPERATOR: choose-operation choose-machine
                       choose-queue-position
      STATUS: posted
      SCHEDULING-DIRECTION: backward
      INITIAL-SEARCH-STATE: {{ INSTANCE search-state
                               OPERATION: 980 }}
   } }}
```

Figure 5-1: Posted Manufacturing Order

The search begins at the specified initial states. The specified operators extend these states. After each application of an operator, the generated states are rated by applying the constraints, and only the best "n" states are kept for the next iteration of operator application. Each state in the path defines one more operation, machine, and queue binding for the order. A complete schedule is defined by the path from the initial state to the end state in search space.

5-3. Constraint Resolution

The key part of the search for a schedule is the application of constraints in rating a search state (partial schedule). 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.

5-3-1. Local Resolution

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 other 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-II filters the set by evaluating each constraint's precondition. The precondition is the final context-sensitive test of a constraints applicability. Only constraints with a true precondition form the final local constraint set.

5-3-2. Global Resolution

Unlike some simple game tree searches, the path which leads to a search state, is as important as the state itself. Local resolution, as defined above, resolves what constraints affect only the state under consideration. But 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.

Constraints are classified into two categories: invariant (e.g., operation preference, queue ordering) and transient (e.g., a due-date or work in process estimator). When ISIS-II rates a state it collects all the invariant constraints along the path from the initial state to the current, and includes them in the rating. ISIS-II also gathers up all the transient constraints, but does not retain duplications. Only the latest instantiation of a transient constraint (closest to the current state) is saved. Transient constraints are estimators of the rating of a partial schedule. Each application of a transient constraint updates its previous application. The union of invariant and transient constraints from the constraint set for the state under consideration.

5-3-3. Relative Resolution

After the constraint set is resolved, each constraint is weighted. The relative importance of a constraint is defined by a scheduling-goal. A scheduling-goal partitions the constraints, assigning a weight to the distributed amongst a partition's members. For example, an order of priority-class "forced outage" would place greater weight on the satisfaction of time constraints such as due-date, and less on constraints such as queue preferences, queue stability, etc.

5-4. Constraint Application

Once the constraint set has been resolved and weighted, ISIS-II derives a rating for the state by computing the weighted average of each constraint's rating of the state. The state stores each constraint's weight and rating at the state for later use in explaining a schedules search path.

Once all the current states have been rated, all but the top "n" states are thrown away and the search is repeated.

5-5. Post-Search Analysis

As the search proceeds, ISIS-II continually tests to see whether the goal state has been generated, and/or the search has died without finding a solution. If either of these has occured, post-search analysis is entered. Post-search analysis is accomplished by a set of rules that analyse the results in order to determine whether:

-the system found a satisfactory solution.

-the system should continue searching if the solution is unsatisfactory.

-another search strategy should be pursued.

Consider the example of mfg-order-xxxx. It is of priority-class "forced outage". A forced outage blade must be shipped by its due date, if not sooner. ISIS-II first attempts to schedule it backwards from its due date. If a feasible schedule is not found, i.e., it runs out of time before it runs out of operations, it then resets the orders scheduling parameters to schedule forward from the "today". If the results of this scheduling attempt are poorly rated, then it marks the order as having failed again, and posts it for re-scheduling. This time, the pre-analysis rules add another operator which searches the shift dimension for a machine.

6. Conclusions

ISIS-II represents an approach to doing constraint-directed reasoning in a very large solution space. It differs from other constraint-based reasoning systems (e.g., Fikes, 1970; Goldstein, 1977; Sussman & Steele, 1980; Stefik, 1981; Waltz, 1975; Zucher, 1976), in the variety of constraints it can represent, and its focus on constraint relaxation as an integral part of the knowledge representation and search process.

ISIS-II is continually being tested on large amounts of data, resulting in the alteration of existing constraints, and the addition of new ones. Its approach has been rated highly by the expert schedulers, and is scheduled to be placed in the factory by September of 1982.

7. References

Fikes R.E., (1970), "REF-ARF: A System for Solving Problems Stated as Procedures", Artificial Intelligence, Vol. 1, pp27-120.

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

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

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

Sussman G.J., and G. L. Steele Jr., (1980), "CONSTRAINTS-A Language for Expressing Almost-Hierarchiacal Descriptions", Artificial Intelligence, Vol. 14, pp1-39.

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 Ambiguitities, In Pattern Recognition and Artificial Intelligence, C. H. Chen (Ed.), New York: Academic Press.