Proceedings of 4th International
Conference on Expert Systems in
Production & Operations Mgmnt
Hilton Head, Sth Carolina,
May 1990.

VARIABLE AND VALUE ORDERING HEURISTICS FOR ACTIVITY-BASED JOB-SHOP SCHEDULING

Norman Sadeh and Mark S. Fox

Center for Integrated Manufacturing Systems
The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

Earlier research in job shop scheduling has demonstrated the advantages of opportunistically combining order-based and resource-based scheduling techniques. In this paper we investigate an even more flexible approach where each activity is considered a decision point by itself. Within this framework, we introduce a probabilistic model that combines both resource-based and order-based insight at the activity level. The model is used to define heuristics that opportunistically select the next decision point on which to focus attention (i.e. *variable ordering* heuristics) and the next decision to be tried at this point (i.e. *value ordering* heuristics). Preliminary experimental results indicate that our variable ordering and value ordering heuristics greatly increase search efficiency. While least constraining value ordering heuristics have been advocated in the literature [Keng 89], our experimental results suggest that other value ordering heuristics combined with our variable-ordering heuristic can produce much better schedules without significantly increasing search.

1.0 Introduction

We are concerned with the issue of how to opportunistically focus an incremental job shop scheduler's attention on the most critical activities[1](*variable ordering* heuristics) and the most promising reservations for these activities (*value ordering* heuristics) in order to reduce search and improve the quality of the resulting schedule.

So called order-based [Fox 83] and resource-based [Adams 88] scheduling techniques have been at the origin of several incremental scheduling schemes. In an order-based approach, each order is considered a single decision point, i.e. orders are prioritized and scheduled one by one. In a resource-based approach, resources are rated according to their projected levels of demand. Resources are then scheduled one by one, starting with the ones that have the highest demand. Order-based scheduling has proven to be a viable paradigm in problems where slack (i.e. precedence constraint interactions) is the dominating factor. On the other hand resource-based scheduling is likely to perform better in situations where resource contention (i.e. resource requirement interactions) is critical. Neither approach is perfect. Indeed a lot of real life scheduling problems contain a mix of critical orders and critical resources. In the past few years it has become clear that in order to perform well in a wider class of problems, schedulers need the ability to opportunistically combine insight from both problem decompositions. The OPIS [Smith 85, Ow 88] scheduler was the first one to combine both approaches by dynamically combining two problem solving perspectives: an order-based perspective and a resource-based perspective. When OPIS identifies a bottleneck resource with a projected demand higher than some threshold, it switches to its resource-based perspective. Otherwise, by default, the system uses an order-based perspective, where orders are scheduled one at a time.

Even such an approach has its shortcomings. Indeed, once a critical order or a critical resource has been identified by OPIS, all the activities within that order or competing for that resource are assumed to be sufficiently

[1] In this section, the criticality of a scheduling entity such as an order, a resource, or an activity, is informally defined as the difficulty in finding a good schedule for that entity.

critical to deserve to be immediately scheduled. This is not necessarily the best possible strategy. Instead some activities within a critical order or contending for a critical resource may not be that critical themselves. This is because, in general, the criticality of an activity should be determined *both* by the order to which it belongs and the resources for which it competes. By scheduling less critical activities before more critical ones, a scheduler will prematurely restrict its remaining options. This in turn will typically translate into poorer schedules and more backtracking (i.e. partial schedules that cannot be completed).

These considerations lead us to investigate a new scheduling framework where the decision points are no longer entire resources or entire orders but instead where each activity is a decision point in its own right. Within this framework, activity criticality accounts for both precedence constraint interactions (i.e., so-called *intra-order* interactions [Smith 85]) and resource requirement interactions (i.e., so-called *inter-order* interactions). By simultaneously accounting for both types of interactions the approach opportunistically combines advantages of both order-based and resource-based scheduling techniques.

In this paper, we introduce a probabilistic model that allows to simultaneously account for both intra-order and inter-order interactions. A similar model was first proposed in [Muscettola 87], where it was used to perform bottleneck analysis given a predetermined variable ordering. Our model does not require such assumptions. Instead we show that it can precisely be used to decide which activity to schedule next (variable ordering heuristic) and which reservation to assign to that activity (value ordering heuristic). Experimental results are presented that indicate that our variable ordering heuristic allows for significant reductions in search. We also compare the performance of three different value ordering heuristics. While least-constraining value ordering heuristics had been advocated in the literature [Keng 89] , our study indicates that it is possible to produce much better schedules without significantly increasing search.

In the next section we describe our model of the job-shop scheduling problem. The following section gives an overview of the activity-based approach to scheduling that we are investigating, and introduces a probabilistic model to account for both intra-order and inter-order interactions. Section 4 presents a variable-ordering heuristic based on this probabilistic model. In section 5 we present three value-ordering heuristics: a least constraining heuristic, a greedy heuristic, and an intermediate heuristic. Preliminary experimental results are reported in section 6. Section 7 discusses these results. Section 8 contains some concluding remarks.

2.0 The Model

Formally, we will say that we have a set of N jobs (i.e. orders) to schedule. Each job has a predefined process plan that specifies a partial ordering among the activities (i.e. operations) to be scheduled. Each activity A_k $(1 \leq k \leq n)$ may require one or several resources R_{ki} $(1 \leq i \leq p_k)$, for each of which there may be several alternatives R_{kij} $(1 \leq j \leq q_{ki})$[2]. We will use st_k, et_k, and du_k to respectively denote A_k's start time, end time, and duration.

We view the scheduling problem as a constraint satisfaction problem (CSP).

The variables of the problem are the activity start times, and the resources allocated to each activity. An activity's end time is defined as the sum of its start time and duration. Each variable has a bounded set of admissible values. For instance, the start time of an activity is always restricted at one end by the order release date and at the other end by the order latest acceptable completion time[3] according to the durations of the activities that precede/follow the activity in the process plan.

We differentiate between two classes of constraints: activity precedence constraints and resource capacity constraints. The activity precedence constraints are the ones defined by the process plans. Our model [Sadeh 88] accounts for all 13 of Allen's temporal constraints [Allen 84]. Capacity constraints restrict the number of reservations of a resource over any time interval to the capacity of that resource. For the sake of simplicity, we will

---

[2]It is important to keep in mind that several activities may require the same resource. For instance if two activities A_1 and A_2 both require a unique resource which has to be R_1, we have $R_{111} = R_{211} = R_1$.

[3]This is not necessarily the order's due date.

assume in this paper that all resources are of unary capacity.

We assume scheduling problems with a discrete time granularity Δ, i.e. activity start times and end times have to be multiples of Δ.

Additionally our model allows for preferences on activity start times as well as on the resources that activities can use. Preferences are modeled with preference functions. These functions map each variable's possible values onto preferences ranging between 0 and 1. Preferences on activity start times allow for the representation of organizational goals such as reducing order tardiness, or reducing inventory (both in-process and finished-goods inventory) [Fox 83, Sadeh 88]. Resource preferences are very useful to differentiate between functionally equivalent resources with different characteristics (e.g. different operating costs). In this paper we assume that the sum of the preference functions defines a (separable) objective function to be maximized.

3.0 The Approach

3.1 An Activity-based Scheduler

In an activity-based approach, each activity is treated as an aggregate variable, or decision point, that is comprised of the activity's start time, and its resources. The schedule is built incrementally by iteratively selecting an activity to be scheduled and a reservation for that activity (i.e. a start time and a set of resources). Every time a new activity is scheduled, new constraints are added to the initial scheduling problem, and propagated. If an inconsistency is detected during propagation, the system backtracks. The process stops either when all activities have been successfully scheduled or when all possible alternatives have been tried without success.

The efficiency of such an incremental approach critically relies on the order in which activities are scheduled and on the order in which possible reservations are tried for each activity. Indeed, because job-shop scheduling is NP-hard, search for a schedule may require exponential time in the worst case. Both empirical and analytical studies of constraint satisfaction problems reported in [Haralick 80, Freuder 82, Purdom 83, Nadel 86a, Nadel 86b, Nadel 86c, Stone 86, Fox 89] indicate however that, on the average, search can significantly be reduced if always focused on the most critical decision points and the most promising decisions at these points. Such techniques are often referred to as variable and value ordering heuristics [Dechter 88].

In this paper we assume that critical activities are the ones whose good (overall) reservations are most likely to become unavailable if one were to start scheduling other activities first. In general reservations may become unavailable because of operation precedence constraints, because of resource capacity constraints, or because of combinations of both types of constraints. Clearly criticality measures are probabilistic in nature, as their computations require probabilistic assumptions on the values that will be assigned later on to each variable (i.e. the reservations that will later on be assigned to each unscheduled activity). In the next subsection we introduce a probabilistic framework that accounts for the interactions of start time and resource preferences induced by both activity precedence and resource capacity constraints. We will use this model throughout the remainder of the paper to define several variable and value ordering heuristics for activity-based scheduling.

3.2 A Probabilistic Framework to Account for Constraint Interactions

In this subsection we outline[4] a probabilistic model that we will use throughout the remainder of the paper to define several variable and value ordering heuristics for activity-based job-shop scheduling. We justify the model by its ability to account for both intra-order and inter-order interactions and by its relatively low computational requirements.

In our model a priori probability distributions are assumed for the possible start times and resources of each

[4] A more detailed description can be found in [Sadeh 88].

unscheduled activity. These probabilities are then refined to account for the interactions induced by the problem constraints (i.e. both intra-order and inter-order interactions). Finally the results of this propagation process are combined to identify critical activities and promising reservations for these activities. In their simplest form the a priori probability distributions are uniform. This amounts to assuming that, a priori, all possible reservations are equally probable. A slightly more sophisticated model consists in *biasing* the a priori distributions towards good values as defined by the preference functions [Sadeh 88]. This allows to give more weight to reservations that are more likely to result in good schedules.

Once the a priori distributions have been built, they can be refined to account for the interactions of the problem constraints. In our model, the propagation is performed in two steps. The probability distributions are first propagated within each order, thereby accounting for intra-order interactions, and then across orders to account for inter-order interactions. Accounting simultaneously for both types of interactions seems indeed very difficult as much from a theoretical point of view as from a purely computational point of view. As a matter of fact the number of ways in which a set of activities can interact is combinatorial in the number of these activities[5]. Instead, by separately accounting for intra-order and inter-order interactions, one greatly reduces the amount of computation to be performed. The experimental results reported at the end of this paper indicate that such two-step approximation is sufficient to guide our scheduler.

Concretely, once the a priori distributions have been generated, our propagation process involves the following two steps:

1.

   a. The a priori start time probability distributions are refined to account for activity precedence constraints. The resulting (a posteriori) probability distributions associate to the possible start times of each activity the probability that these start times will be tried by the scheduler and will not result in the violation of an activity precedence constraint. These a posteriori start time distributions can be normalized to express that each activity will occur exactly once, and hence will start exactly once.

   b. For each resource requirement R_{ki} of each activity A_k, and for each resource alternative R_{kij} to fulfill R_{ki}, we compute the probabilistic demand D_{kij} of A_k for R_{kij} as a function of time. This probability is obtained using A_k's normalized a posteriori start time distribution and the a priori probability that A_k uses R_{kij} to fulfill its requirement R_{ki}. Hence $D_{kij}(t)$ represents the probabilistic contribution of A_k to the demand for R_{kij} at time t, if activity reservations were only checked for consistency with respect to the activity precedence constraints. Later on we will refer to $D_{kij}(t)$ as A_k's (probabilistic) individual demand for R_{kij} at time t.

2. Finally the individual demand densities of all activities are aggregated (i.e. summed at each point in time) to reflect the probabilistic demand for each resource in function of time. The resulting aggregate demand densities may get larger than one over some intervals of time, as the individual demand densities from which they originate have not been checked for consistency with respect to the capacity constraints. High demand for a resource over some time interval indicates a critical resource/time interval pair, which requires prompt attention from the scheduler. This is the basis to the variable-ordering heuristic presented in this paper. Additionally, we also record the number of activities contributing to the aggregate demand for a resource as a function of time. This provides for more accurate value ordering heuristics.

[5]In any realistic problem, Monte Carlo simulation would indeed require tremendous amounts of computations if one were to simultaneously account for all the activities and all the constraints. This is because the probability of randomly generating a schedule for all the activities, that satisfy all activity precedence and resource capacity constraints, is in general extremely small.

Notations

In the remainder of the paper the following notations will be used:

- $P^{PRIOR}(st_k=t)$ will denote the a priori probability that A_k will be scheduled to start at time t,
- $P^{POST}(st_k=t)$ will be the a posteriori probability that A_k starts at time t, i.e. after accounting for activity precedence constraints,
- $P_N^{POST}(st_k=t)$ represents the same probability distribution after it has been normalized to express that A_k will start exactly once,
- $D_{kij}(t)$ represents A_k's individual demand for R_{kij} at time t
- $D_{R_{kij}}^{aggr}(t)$ will denote the aggregate demand for R_{kij} at time t. and
- $n_{R_{kij}}(t)$ will denote the number of activities contributing to the aggregate demand $D_{R_{kij}}^{aggr}(t)$.

4.0 ARR: A Variable Ordering Heuristic Based on Activity Resource Reliance

ARR, the variable ordering heuristic that we study in this paper, consists in looking for the resource/time interval pair that is the most contended for and the activity that relies most on the possession of that resource over that time interval. This activity is selected as the most critical one and hence is the next one to be scheduled.

The intuition behind this heuristic is the following. If activities that critically rely on the possession of highly contended resources were not scheduled first, it is very likely that, by the time the scheduler would turn its attention to them, the reservations that are the most appropriate for these activities would no longer be available.

The aggregate demand densities introduced in subsection 3.2 are used to identify the most demanded resource/time-interval pair. The activity that contributes most to the demand for the resource over the time interval (i.e. the activity with the largest individual demand for the resource over the time interval) is interpreted as the one that relies most on the possession of that resource. Indeed the total demand of an activity A_k for one of its resource requirement R_{ki} is equal to A_k's duration and is distributed over the different alternatives, R_{kij}, for that resource, and over the different possible times when A_k can be carried out. Consequently activities with a lot of slack or several resource alternatives tend to have fairly small individual demand densities at any moment in time. They rely less on the possession of a resource at any moment in time than activities with less slack and/or fewer resource alternatives. This allows ARR to account not only for inter-order interactions but also for intra-order interactions.

The advantage of this heuristic lies in its simplicity. In practice, its computational cost is significantly lower than that of Keng's heuristic [Keng 89], which requires to inspect all remaining reservations of all unscheduled activities. A possible drawback of the ARR heuristic is that it only considers resource reliance with respect to the resource/time interval that is the most contended for in the current search state. We have experimented with more sophisticated variable ordering heuristics such as the one described in [Keng 89]. As indicated by the experiments reported in this paper, the gain in search efficiency provided by these more sophisticated heuristics, if any, is usually not worth the overhead in computation.

5.0 Three Value Ordering Heuristics

In the experiments that we ran, we considered the following three value-ordering heuristics:

5.1 LCV: A Least Constraining Value Ordering Heuristic

Least constraining value ordering heuristics are known for being very good at reducing search in Constraint Satisfaction Problems [Haralick 80, Dechter 88]. [Keng 89] describes one such heuristic for job-shop scheduling problems. Below we describe LCV, a least constraining value ordering heuristic based on the probabilistic model of

ection 3.

LCV is a least constraining value ordering heuristic where every reservation $\{\{st_k=t, R_{k1j_1}, R_{k2j_2},\ldots, R_{kp_kj_{kp_k}}\}\}$, for an activity A_k, is rated according to the probability $RESERV\text{–}AVAIL(st_k=t, R_{k1j_1},\ldots,R_{kp_kj_{P_k}})$ that it would not conflict with another activity's reservation, if one were to first schedule all the other remaining activities. Reservations with large such probabilities are the ones that participate in a large number of schedules compatible with the current partial schedule. They are least constraining reservations. LCV selects the reservation with the highest probability $RESERV\text{–}AVAIL(st_k=t, R_{k1j_1},\ldots,R_{kp_kj_{P_k}})$.

In our model, we express $RESERV\text{–}AVAIL(st_k=t, R_{k1j_1},\ldots,R_{kp_kj_{P_k}})$ as the product of the probability that $st_k=t$ will not result in the violation of an activity precedence constraint and the conditional probability that each resource $R_{k1j_1}, R_{k2j_2},\ldots, R_{kp_kj_{P_k}}$ will be available between t and $t+du_k$, given that $st_k=t$ does not result in the violation of an activity precedence constraint:

$$RESERV\text{–}AVAIL(st_k=t, R_{k1j_1},\ldots,R_{kp_kj_{P_k}}) = \frac{P^{POST}(st_k=t)}{P^{PRIOR}(st_k=t)} \times \prod_{R_{kij} \in \{R_{k1j_1},\ldots,R_{kp_kj_{P_k}}\}} RESOURCE\text{–}AVAIL(R_{kij}, t, t+du_k)$$

where $RESOURCE\text{–}AVAIL(R_{kij}, t, t+du_k)$ is the conditional probability that R_{kij} will be available between t and $t+du_k$, given that $st_k=t$ does not result in the violation of an activity precedence constraint.

Assuming that each of the $(n_{R_{kij}}(\tau)-1)$ other activities competing for R_{kij} at time τ equally contributes to the demand $D^{aggr}_{R_{kij}}(\tau)-D_{kij}(\tau)$, we can approximate the (conditional) probability that R_{kij} will be available at some time τ for activity A_k as the probability that none of the other $(n_{R_{kij}}(\tau)-1)$ activities uses that resource at that time, namely:

$$(1-\frac{D^{aggr}_{R_{kij}}(\tau)-D_{kij}(\tau)}{n_{R_{kij}}(\tau)-1})^{(n_{R_{kij}}(\tau)-1)} \qquad (1)$$

When approximating $RESOURCE\text{–}AVAIL(R_{kij}, t, t+du_k)$, one has to be careful not to come up with too pessimistic an estimate. Indeed it is tempting to assume that the (conditional) probability that R_{kij} will be available for A_k between t and $t+du_k$ is given by the product of (1) over all possible start times τ between t and $t+du_k$ (as determined by the granularity Δ of the problem). In general, this approximation is too pessimistic, as it assumes that the activities contributing to $D^{aggr}_{R_{kij}}(\tau)$ have a duration equal to Δ, i.e. that these activities can possibly require R_{kij} at time τ without requiring it at $\tau-\Delta$ or $\tau+\Delta$. Instead, in order to account for the duration of these activities, we will assume that each resource R_{kij} is subdivided into a sequence of buckets of duration $AVG(du)$, where $AVG(du)$ is the average duration of the activities competing for R_{kij}. Consequently $RESOURCE\text{–}AVAIL(R_{kij}, t, t+du_k)$ is given by the probability that A_k can secure a number of buckets equal to its duration, which is approximated as:

$$RESOURCE\text{–}AVAIL(R_{kij}, t, t+du_k) = \{(1-\frac{AVG(D^{aggr}_{R_{kij}}(\tau)-D_{kij}(\tau))}{(AVG(n_{R_{kij}}(\tau))-1)})\}^{\frac{du_k \times (AVG(n_{R_{kij}}(\tau))-1)}{AVG(du)}}$$

where $AVG(D_{kij}(\tau)-D^{aggr}_{R_{kij}}(\tau))$ and $AVG(n_{R_{kij}}(\tau))$ are respectively the averages of $D_{kij}(\tau)-D^{aggr}_{R_{kij}}(\tau)$ and $n_{R_{kij}}(\tau)$ taken between t and $t+du_k$

5.2 GV: A Greedy Value Ordering Heuristic

The second value ordering heuristic that we tested simply consists in ranking the possible reservations of an activity according to their direct contributions to the objective function (or combined preferences), i.e. according to the sum of their start time and resource preferences. Later we refer to this greedy value ordering heuristic as GV.

5.3 INT: An Intermediate Value Ordering Heuristic

Finally the third value ordering heuristic that we used combines features from the previous two. INT is a value ordering heuristic that rates each possible reservation $\{\{st_k=t, R_{k1j_1}, R_{k2j_2},..., R_{kp_kj_{kp_k}}\}\}$ according to the product of $RESERV\text{-}AVAIL(st_k=t, R_{k1j_1},...,R_{kp_kj_{p_k}})$ with the combined preference of that reservation. In other words, INT rates a reservation according to the product of the ratings assigned by GV and LCV to that reservation. As a consequence INT is a heuristic that looks for reservations that maximize the activity's local preferences while leaving room to other unscheduled activities for selecting good reservations as well.

6.0 Preliminary Experimental Results

We have tested the different heuristics described in this paper as well as multiple other ones on several hundred scheduling problems involving up to a hundred activities. In this section, we present a set of experiments that compares the performance of our variable ordering heuristic ARR together with the three value ordering heuristics LCV, GV, and INT. Additionally we have also included the performance of the variable and value ordering heuristics described in [Keng 89], which we refer to as SMU. The SMU heuristics have been reported to achieve high search efficiency for job-shop scheduling and hence constitute a good benchmark to compare the performance of our heuristics. The experiments were run on a set of 20 randomly generated scheduling problems. The set was comprised of 5 problems with 25 activities (5 orders of 5 activities each), 5 with 50 activities (10 orders of 5 activities each), 5 with 75 activities (15 orders of 5 activities each), and 5 with 100 activities (20 orders of 5 activities each). Each problem involved 5 resources. For each problem, process plans were randomly drawn from a set of two process plans in order to have a high correlation in their resource requirements, and hence higher resource contention. Activity durations were randomly drawn from distributions that guaranteed that at least one out of the five resources would be a main bottleneck. In order to further complicate the problems, all orders were given identical release dates and latest acceptable completion dates. Order due dates were randomly drawn between the common release date and latest acceptable completion date. Finally start time preferences were added to reduce order tardiness and inventory (i.e. both in-process inventory and finished-goods inventory).

All heuristics were run in a modular testbed that allows for sharing all common functions (e.g. consistency labeling module, backtracking module, etc), bypassing unnecessary functions whenever possible (e.g. bypassing the probabilistic computations when using *SMU*) or using alternative functions (e.g. the variable and value ordering heuristics). In all cases search was stopped if it required more than 1000 search states.

For each scheduling problem, the heuristics were rated along the following dimensions:

- Search efficiency: the ratio of the number of activities to be scheduled over the total number of search states that were explored. In the absence of backtracking, only one search state is generated for each activity, and hence search efficiency is equal to 1.
- Number of experiments solved in less than 1000 search states each (out of 20).
- Average CPU time (in seconds) to *successfully* schedule an activity on a VAX8800 running Knowledge Craft on top of Common Lisp. This measure was obtained by simply dividing the total CPU time by the number of activities to be scheduled.
- Schedule value: this is a normalized version of the objective function. It indicates how well the heuristic was able to satisfy the preference functions. In the absence of constraints, the maximum schedule value would be equal to 1. Typically, because of precedence and capacity constraints, the optimal schedule value is lower than 1.

- Order tardiness
- Order inventory: computed as the some of order earliness (finished-goods inventory) and order flowtime (in-process inventory).

The last three measures (schedule value, order tardiness, and order inventory) were computed for the 18 experiments (out of 20) solved by all four heuristic combinations in less than 1,000 search states.

The table in Figure 6-1 summarizes the average performance of each of the four heuristics with respect to the six measures defined above. Standard deviations are given between parentheses.

	ARR&LCV	ARR&INT	ARR&GV	SMU
Search Efficiency	0.89 (0.27)	0.94 (0.23)	0.77 (0.39)	0.95 (0.22)
Nb. exp. solved in less than 1000 states	20	19	18	19
CPU sec. per activity	17.97 (12.98)	18.49 (13.38)	21.71 (12.13)	23.02 (17.36)
Schedule Value	0.47 (0.04)	0.73 (0.05)	0.86 (0.10)	0.41 (0.05)
Order Tardiness	821 (391)	542 (244)	241 (119)	756 (337)
Order Inventory	1708 (673)	1256 (441)	702 (125)	1523 (568)

Figure 6-1: Comparative study of four different heuristics.
Standard deviations appear between parentheses.

7.0 Discussion

The results reported in Figure 6-1 indicate that our variable ordering heuristic, ARR, allowed for a search efficiency comparable to that of the SMU heuristic, while requiring significantly less time. In particular ARR&LCV required 17.97 seconds per activity whereas SMU required 23.02 sec (a speed-up of 22%). On a set of more difficult experiments with two major bottleneck resources, the efficiency of SMU dropped to 77%, whereas that of ARR&LCV only dropped to 86%. On these more difficult problems, ARR&LCV achieved a remarkable 100% speed-up compared to SMU.

Our results also indicate that least constraining value ordering heuristics such as SMU or ARR&LCV, although very good at reducing search, tend to produce fairly poor schedules. Instead ARR&INT achieved a search efficiency almost identical to that of SMU, while allowing for a reduction of almost 30% in order tardiness and 20% in inventory. ARR&GV allowed for even better schedules, though at the expense of search efficiency. On more difficult problems, the efficiency of ARR&GV tends to degrade even further, whereas ARR&INT still fares very well. These results reflect a tradeoff between search and the quality of the resulting schedules.

8.0 Concluding Remarks

In this paper, we have investigated an activity-based approach to scheduling. Because of its greater flexibility, such an approach is expected to allow for the construction of better schedules than approaches using order-based or resource-based scheduling or even combinations of the two. The price to pay for this flexibility is the potential overhead involved in the selection of the next decision point on which to focus attention, since there are a lot more possibilities. For this reason, it is particularly important to come up with variable and value ordering

heuristics that require as little computation as possible.

In this paper, we have presented a probabilistic framework that allows for the definition of such heuristics. Our model accounts both for intra-order and inter-order interactions, and allows for the definition of different variable and value ordering heuristics. We have presented a simple variable-ordering heuristic, ARR, that looks for the most contended resource/time interval pair and the activity that relies the most on the possession of that resource/time interval. Our experiments indicate that this heuristic allows for very high search efficiency, while being much faster than that described in [Keng 89]. Our experiments also indicate that least constraining value ordering heuristics such as the one advocated in [Keng 89] are not the only viable way to maintain search at an acceptable level. Instead we have shown that other value ordering heuristics , when coupled with our variable ordering heuristic, produced much better schedules without significantly increasing search.

Acknowledgement

This research was supported, in part, by the Defense Advance Research Projects Agency under contract #F30602-88-C-0001, and in part by a grant from McDonnell Aircraft Company.

References

[Adams 88] J. Adams, E. Balas, and D. Zawack.
The Shifting Bottleneck Procedure for Job Shop Scheduling.
Management Science 34(3):391-401, 1988.

[Allen 84] J.F.Allen.
Towards a General Theory of Action and Time.
Artificial Intelligence 23(2):123-154, 1984.

[Dechter 88] Rina Dechter and Judea Pearl.
Network-Based Heuristics for Constraint Satisfaction Problems.
Artificial Intelligence 34(1):1-38, 1988.

[Fox 83] M. Fox.
Constraint-Directed Search: A Case Study of Job-Shop Scheduling.
PhD thesis, Department of Computer Science, Carnegie-Mellon University, 1983.

[Fox 89] Mark S. Fox, Norman Sadeh, and Can Baykan.
Constrained Heuristic Search.
In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 309-315. 1989.

[Freuder 82] E.C. Freuder.
A Sufficient Condition for Backtrack-free Search.
Journal of the ACM 29(1):24-32, 1982.

[Haralick 80] Robert M. Haralick and Gordon L. Elliott.
Increasing Tree Search Efficiency for Constraint Satisfaction Problems.
Artificial Intelligence 14(3):263-313, 1980.

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

[Muscettola 87] Nicola Muscettola, and Stephen Smith.
A Probabilistic Framework for Resource-Constrained Muti-Agent Planning.
In *Proceedings of the Tenth International Conference on Artificial Intelligence*, pages 1063-1066. 1987.

[Nadel 86a] B.A. Nadel.
The General Consistent Labeling (or Constraint Satisfaction) Problem.
Technical Report DCS-TR-170, Department of Computer Science, Laboratory for Computer Research, Rutgers University, New Brunswick, NJ 08903, 1986.

[Nadel 86b] B.A. Nadel.
Three Constraint Satisfaction Algorithms and Their Complexities: Search-Order Dependent and Effectively Instance-specific Results.
Technical Report DCS-TR-171, Department of Computer Science, Laboratory for Computer Research, Rutgers University, New Brunswick, NJ 08903, 1986.

[Nadel 86c] B.A. Nadel.
Theory-based Search-order Selection for Constraint Satisfaction Problems.
Technical Report DCS-TR-183, Department of Computer Science, Laboratory for Computer Research, Rutgers University, New Brunswick, NJ 08903, 1986.

[Ow 88] Peng Si Ow and Stephen F. Smith.
Viewing Scheduling as an Opportunistic Problem-Solving Process.
Annals of Operations Research 12:85-108, 1988.

[Purdom 83] Paul W. Purdom, Jr.
Search Rearrangement Backtracking and Polynomial Average Time.
Artificial Intelligence 21:117-133, 1983.

[Sadeh 88] N. Sadeh and M.S. Fox.
Preference Propagation in Temporal/Capacity Constraint Graphs.
Technical Report CMU-CS-88-193, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, 1988.
Also appears as Robotics Institute technical report CMU-RI-TR-89-2.

[Smith 85] Stephen F. Smith and Peng Si Ow.
The Use of Multiple Problem Decompositions in Time Constrained Planning Tasks.
In *Proceedings of the Ninth International Conference on Artificial Intelligence*, pages 1013-1015. 1985.

[Stone 86] Harold S. Stone and Paolo Sipala.
The average complexity of depth-first search with backtracking and cutoff.
IBM Journal of Research and Development 30(3):242-258, 1986.

Table of Contents

List of Figures