Heuristics for Constraint-Directed Scheduling with Inventory

J. Christopher Beck
ILOG, S.A.
9, rue de Verdun, B.P. 85
F-94253 Gentilly CEDEX
FRANCE
cbeck@ilog.fr

Abstract

Despite the importance of the management of inventory in industrial scheduling applications, there has been little research that has addressed reasoning about inventory directly as part of a scheduling problem. In this paper, we represent inventory, inventory storage constraints, and inventory production and consumption in a constraint-directed scheduling framework. Inventory scheduling is then used to investigate heuristic commitment techniques based on the understanding and the exploitation of problem structure. A technique for the estimation of probability of breakage for resource and inventory constraints is presented together with a heuristic commitment technique based on the estimate of constraint criticality. It is empirically demonstrated that a heuristic commitment technique that exploits dynamic constraint criticality achieves superior overall performance.

Introduction

The management of inventory, its storage, production, and consumption, represents the core function of a manufacturing organization. Little scheduling research, however, has specifically addressed inventory. While there are a wide variety of commercial scheduling systems that deal with inventory, the techniques used in these systems remain proprietary.

The central contribution of this paper is the expansion of a structural analysis technique (called *texture measurements*) to account for a richer constraint representation and the application of texture measurements to the problem of scheduling with inventory constraints. A broader contribution of this paper is the demonstration of the ease of extension of the constraint-directed approach to problem solving to problems with novel characteristics.

This paper is organized as follows: we first define the inventory scheduling problem investigated in this paper and present the methodology for the construction of problem instances. We then describe two areas of previous work: investigations of similar scheduling problems and work on texture-based heuristics on which the inventory heuristics in this paper are based. A detailed presentation of the inventory representation and the heuristic commitment techniques themselves then follows. We evaluate the heuristic commitment techniques in two experimental conditions, the results of which are then discussed.

A Simple Inventory Scheduling Problem

An $n \times m$ inventory scheduling problem consists of n jobs and m resources. Each job is composed of m activities, each using a different resource. Each activity, A_{ij}, in job, j:

- has a constant duration, dur_{ij}.
- uses one resource, R_{ij}, with no interruption, for its entire duration.
- is completely ordered with the other activities in job j. If A_{ij} is before A_{kj} in the complete ordering, A_{ij} must finish executing before A_{kj} can begin executing.
- may consume some amount of one or more inventories. Consumption is assumed to happen instantaneously at the start of execution.
- may produce some amount of one or more inventories. Production is assumed to happen instantaneously at the end of execution.

In addition to the precedence constraints among activities in the same job, there are two additional types of constraints:

1. Unary resource constraints – each resource can be used by at most one activity at any time point.
2. Inventory constraints – each inventory has a maximum and minimum constraint which specify, respectively, the maximum and minimum amount of each type of inventory that can exist at any time point.

The jobs, activities, activity characteristics (duration, resource usage, inventory production/consumption), resources, and inventories are all given in the problem definition. A solution consists of a sequence of activities on each resource such that all constraints (precedence, resource, and inventory) are satisfied.

This problem definition represents the minimal addition of inventory representation to the job shop scheduling problem (Garey and Johnson, 1979; Blazewicz et al., 1996). While real-world inventory problems contain more complex inventory requirements (Beck, 1999), the relative lack of research literature addressing such requirements mandates a simple problem definition so that we can begin to systematically investigate inventory scheduling.

Generating Inventory Problems

Two variations of the simple inventory scheduling problem are investigated in this paper. In the *one-stage* problems, only raw material inventory are consumed by the activities and only finished goods inventory are produced. *Two-stage* problems add work-in-process inventory: an inventory produced by one job may be required by a subsequent job. These two types of problems allow us to investigate inventory scheduling heuristics as the complexity of the inventory relationship changes: in one-stage problems, the inventory requirements result in otherwise unrelated activities competing for a limited pool of raw materials while in the two-stage problems, activities may also be related through production/consumption requirements. Such requirements are common, especially in continuous manufacturing applications (*e.g.,* activities in stage k+1 consume inventory produced by stage k). The two-stage inventory problems allow us to begin to investigate the inter-job constraints engendered by such process models.

One-Stage Inventory Problems

A single raw material inventory and a single finished goods inventory is associated with each job. This gives the problem a total of 2n inventories. Minimally, the first activity in each job consumes the corresponding raw material and the final activity produces the corresponding finished good. Further inventory interactions are added by specifying, c, the number of consumptions of a raw material inventory in each job. The activities that consume and the raw materials that are consumed are randomly selected with uniform probability and replacement. The only restriction on a single activity is that each consumption must be of a different inventory. The maximum value for c in an $n \times m$ problems is nm which occurs when each activity consumes each of the raw materials.

Given an $n \times m$ job shop problem, we calculate the lower bound on the makespan as described by (Taillard, 1993). The scheduling horizon, then, is based on a *makespan factor* multiplied by the lower bound calculation.

Supply and Demand Events. The raw materials must be introduced into the problem via supply events, while the finished goods must have corresponding demand events. Both these types of events are modeled using activities of 0 duration with predefined start-times. The amount of a raw material RM_i supplied is found by summing the amount of all consumers of RM_i. The amount of RM_i produced by supply events is then calculated by multiplying this sum by a *supply factor.* For all the problems examined here, we set the supply factor to be 1.2. Future work will examine the manipulation of the supply factor.

The time of the supply events for RM_i is determined by calculating, $maxST(RM_i)$, the latest time that a consumer of RM_i can begin execution. This time is found by calculating the minimum tail (Carlier and Pinson, 1994) of all consumers of RM_i and subtracting that value (weighted by the makespan factor) from the length of the scheduling horizon as shown in Equation (1).

$$maxST(RM_i) = horizon - mkspFactor \times minTail(RM_i) \quad (1)$$

The supply of raw material is created via five supply events (each contributing approximately equal amounts of inventory) whose times are evenly distributed on the temporal interval [0, $maxST(RM_i)$]. We choose to use five supply events to mimic real-world factory situations where it is common to have a predefined, regular delivery schedule for raw materials inventory.

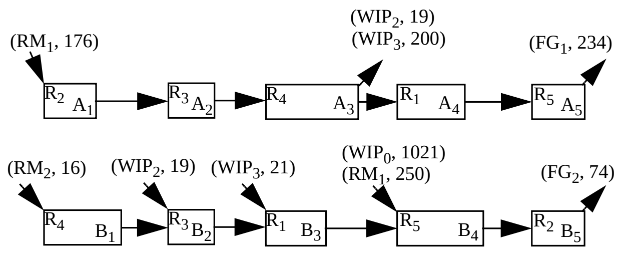

Figure 1. Two Jobs from a 5×5 Two-Stage Inventory Problem.

There is a single demand event for each finished good. It occurs at the end of the horizon and demands the total amount of a finished good produced in the problem.

Inventory Constraints. For all inventories, the minimum constraint is an inventory level of 0. For a finished good, FG_i, the maximum constraint is equal to the total amount of FG_i produced in the problem. Since the total amount of a finished good that is produced is equal to the maximum limit, the maximum constraint on finished goods does not constrain the problem.

For a raw material, RM_i, the maximum constraint is equal to 75% of the total amount of RM_i produced by supply events.

Two-Stage Inventory Problems

We generate two-stage problems by combining two one-stage problems. There are three components to the combination process: inventory, resource, and temporal.

- Inventory – given the two one-stage problems, P_1 and P_2, the supply events in P_2 are replaced with production by activities in P_1. The activities that produce each work-in-process inventory are chosen randomly with uniform probability from the activities in P_1. The only restriction is that each inventory is produced by five different activities.
- Resources – only the resources from P_1 exist in the combined problem. The activities from P_2 are randomly assigned to execute on a P_1 resource with the usual job shop restriction that each activity in a job executes on a different resource.
- Temporal – we find the lower bound on makespan by summing the durations of the activities on each resource. This lower bound is then multiplied by the makespan factor to generate the length of the scheduling horizon. Each demand event is scheduled to occur at the end of the horizon regardless of whether the finished good inventory originally came from P_1 or P_2. Each supply event occurs as described in the single-stage problems; however, the timing of each event is changed to reflect the makespan of the two-stage problem.

For example, Figure 1, displays two jobs from a 5×5 two-stage problem. The second job, B, is from the second stage as it consumes work-in-process inventory (e.g., WIP_2) that is produced by a first-stage job such as A.

The combination of two $n \times m$ problems results in a single problem with m resources, $4n$ inventories (n raw materials, n intermediate inventories, and $2n$ finished goods), and $2n$ jobs each with m activities. There are $5n$ supply events (five for each raw material in P_1) and $2n$ demand events (one for each finished good in P_1 and P_2).

Background

Given the simple inventory problem model described above, we now examine previous work that has considered similar problems as well as work on texture measurements which forms the basis for the heuristic commitment techniques for inventory introduced in this paper.

Inventory Scheduling

The published techniques for inventory scheduling typically make use of constraint propagation to maintain inventory constraints while leaving the heuristic techniques to concentrate on scheduling the re-usable resources.

In CHIP (Simonis and Cornelissens, 1995), the cumulative constraint is used to represent inventory as a reusable resource. An activity which produces an inventory at some time, t_1, is represented as an activity that uses the corresponding resource from time 0 to time t_1. At t_1, the activity ends and the resource is released for use by other activities. Similarly, an activity which consumes an inventory at some time, t_2, is represented as an activity which uses the corresponding resource from t_2 to the end of the scheduling horizon. The cumulative constraint specifies that the sum of all the activities using the resource at any time point must be less than the maximum inventory level. The minimum inventory constraint can also be represented with a separate cumulative constraint with a slight modification in the modeling of producers and consumers.

In ILOG Scheduler, one method of inventory modeling is the use of a time-table mechanism (Le Pape, 1994b; Le Pape, 1994a) in which each inventory has a time-table defining the time-varying minimum and maximum capacity constraints. Activities produce and consume inventory, and propagation is done through the time-tables to prune start times that are not consistent with the inventory constraints.

The KBLPS distribution planner (Saks, 1992) treats the initial inventory and subsequent incoming supplies as separate discrete quantities. All the activities in one order are scheduled before moving to the next order. This enables the algorithm to use texture measurements to identify the most critical resource or inventory, and to schedule the order which relies most on that resource or inventory.

In CHIP and ILOG Scheduler, the primary use of the inventory modeling is for propagation. With such an approach, traditional scheduling algorithms can be used to assign start times to activities, while the inventory propagation maintains the inventory constraints. A weakness of this approach is that no heuristics directly examine the inventory constraints, even if inventory constraints are the major challenge in solving a problem. The KBLPS model, in contrast, directly represents and reasons about inventory as part of the heuristic commitment technique. However, the approach depends on the order-based problem decomposition and the scheduling of all inventory transitions within an order.

Our approach to inventory scheduling is to extend texture measurements to evaluate the criticality of both resource and inventory constraints. This criticality measurement can then be used as a basis for the dynamic focus of the heuristic commitment technique enabling it to identify the most critical constraint in each search state and then to find a commitment that will tend to reduce that criticality.

Texture Measurements

Our approach to heuristic search rests on the problem structure hypothesis (Simon, 1973) which states that an understanding of the problem structure is central to high-quality heuristic search. With a constraint graph representation, understanding of the problem structure arises from the analysis of the constraint graph. We use texture measurements (Fox, 1983; Fox et al., 1989; Sadeh, 1991) to distill information from the constraint graph and then base our heuristic decision-making on the distilled information.

A *texture measurement* is an analysis of the constraint graph underlying a problem state that distills information that can be used by a heuristic commitment technique (Fox et al., 1989). For example, *contention* (Sadeh, 1991) is the extent to which variables linked by a disequality constraint compete for the same value. In the context of scheduling, contention is the extent to which activities compete for the same resource over the same time interval.

The foundation of the contention measurement and later extensions (Muscettola, 1992; Beck et al., 1997b) is a probabilistic estimate of each activity's demand for a resource over time. The individual activity demands are summed to form the aggregate demand over time for each resource and the aggregate demand is then used to dynamically identify the critical resource and time point. The heuristic uses the criticality information to focus on the most important part of the problem in each search state.

In this paper, we measure the criticality of a constraint by its probability of breakage as estimated by the VarHeight texture measurement, previously applied to unary capacity resources (Beck et al., 1997a). VarHeight aggregates the individual expected value and variance of each activity's demand for a resource into aggregate expected value and aggregate variance curves. The aggregate curves are then used to estimate the probability of breakage of the resource constraint at each time point.

In extending VarHeight to inventory constraints, we modify the individual expected value and variance curves. An activity that produces or consumes inventory makes a positive or negative contribution to the level of that inventory. Assuming that each of the remaining start times is equally likely to be assigned and considering inventory I, a time point t, and an activity A, we associate a random variable X with the contribution that A has to I at time t. The

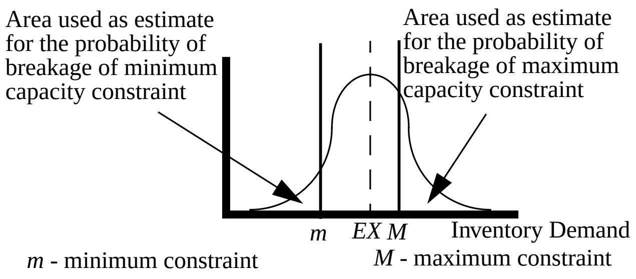
Area used as estimate for the probability of breakage of minimum capacity constraint
Area used as estimate for the probability of breakage of maximum capacity constraint
m
EX
M
Inventory Demand
m - minimum constraint
M - maximum constraint

Figure 2. Calculating the Probability of Breakage at Time *t* with VarHeight.

domain of X is $\{0, AMT_A(I)\}$ where $AMT_A(I)$ is the total amount of inventory I that A produces or consumes. If A is a consumer, $AMT_A(I) < 0$.The expected value for X at time, t, $EX_A(t)$, is $ID(A, I, t)$. If A is a production activity, ID is calculated as follows, for all $eft_A \le t \le lft_A$ (where eft_A and lft_A are, respectively the earliest and latest finish times of A and STD_A is the domain of the start time variable of A):

$$ID(A, I, t) = AMT_A(I) \times \frac{t - eft_A + 1}{|STD_A|} \qquad (2)$$

If A is a consumption activity, ID is calculated the same way, relative to the start time window. That is, for all $est_A \le t \le lst_A$ (where est_A and lst_A are the earliest and latest start times of A):

$$ID(A, I, t) = AMT_A(I) \times \frac{t - est_A + 1}{|STD_A|} \qquad (3)$$

We calculate the variance of X at time, t, $VX_A(t)$, as follows (see (Beck, 1999) for the full derivation):

$$VX_A(t) = EX_A(t) \times (AMT_A(I) - EX_A(t)) \qquad (4)$$

Given these definitions, we can calculate $EX_A(t)$ and $VX_A(t)$ for all activities, A, at time point t.

The aggregation of the individual demands on inventory is done exactly as the aggregation on resources described in (Beck et al., 1997a). In particular, the same assumptions and same time-complexity apply: $O(mn \log n) + O(mn)$ (where n is the maximum number of activities on a resource and m is the number of resources).

With this formulation, the VarHeight texture measurement can be used to estimate the probability of breakage (over time) of resource and inventory constraints as illustrated in Figure 2. (Note that the normal distribution is an assumption discussed in (Beck et al., 1997a)).

Texture-Based Heuristics

Given an estimate of the criticality of each constraint at each time point, we then use a heuristic commitment technique that makes use of the criticality information. As presented in more detail below, the VarHeight heuristic has different behavior depending on what type of constraint is found to be most critical. If a resource constraint is most critical, the heuristic attempts to sequence the two activities that contribute most to the criticality. In contrast, if an inventory minimum constraint is most critical, the VarHeight heuristic identifies the consumer that contributes most to that criticality and constrains it to consume inventory from an available producer. A similar commitment is made if an inventory maximum constraint is most critical. In all cases, the intuition behind the heuristic is the same: given the texture measurement information that identifies the most critical constraint, the heuristic makes a commitment that will tend to reduce that criticality.

Inventory Representation

Our inventory representation is a generalization of the resource representation in the ODO scheduler (Beck et al., 1998; Beck, 1999). Each activity has one or more inventory requirements, each represented as a variable that takes the value of an inventory. For each requirement, there is a variable corresponding to the amount of the inventory that the activity produces or consumes. Consumption is assumed to take place instantaneously at the start of an activity while production occurs instantaneously at the end of an activity. The inventory constraints are represented analogously to resource capacity constraints. A maximum constraint defines the highest inventory level allowable at any time point while a minimum constraint defines the lowest inventory level allowable.

The *inventory termination criteria* are a set of conditions under which it is guaranteed that the constraints on an inventory will be satisfied in all subsequent search states. The standard, and usually implicit, termination criterion in a constraint satisfaction problem is that all variables are assigned a value and all constraints are satisfied. Because we are not simply assigning start times to activities (see below), we require that the termination criteria be made explicit. Our inventory termination criteria are that the lower bound on inventory level is greater than or equal to the minimum constraint and the upper bound is less than or equal to the maximum constraint. If both conditions are satisfied, no further commitments are required to guarantee that the inventory constraints are satisfied. This termination criteria requires the calculation and maintenance of the upper and lower bounds on the inventory level for each inventory. An efficient algorithm is presented in (Beck, 1999).

As noted above, much of the previous work on inventory scheduling uses standard scheduling commitments (*e.g.,* start time assignment) and depends on propagation to detect dead-ends and prune alternatives for inventory activities. Start time assignments, however, do not address the root of inventory criticality. If a minimum inventory constraint is critical, for example, there is a danger that some consumer will be scheduled such that no producer can execute to supply its required inventory. The problem is not the assignment of start times, but the ordering of a producer-consumer pair. Our inventory commitment, therefore, consists of a precedence constraint specifying that the producer will occur before the consumer as well as a constraint specifying the amount of inventory produced by the producer that is consumed by the consumer. We include a specific amount of inventory to improve the structural information on which we make inventory commitments: knowing how much of the inventory produced by an activity has already been assigned to be consumed by a particular consumer significantly aids in the identification of critical producers and consumers.

Simply matching consumers with producers does not guarantee that the termination criteria for an inventory (or a dead-end) will be met. It may be the case that we have to further constrain the linked producer-consumer pairs to execute within a certain time interval of each other. In the worst case, in fact, it may be necessary to actually assign start times to some activities in order to guarantee the satisfaction of inventory constraints. To address such situations, we propose three types of commitments to reduce inventory criticality: producer/consumer commitments, producer/consumer interval commitments, and start time commitments.

Producer/Consumer Commitments

We use the notation $(P \rightarrow C, 25)$ to indicate a producer/consumer commitment between activities P and C. Specifically, this notation indicates that C must start at or after the end of P and that 25 units of the inventory produced by P are consumed by C. P is not constrained to *only* produce 25 units of inventory, nor is C constrained to *only* consume 25 units: each may be linked with other consumers and producers, respectively. When all of the inventory that is produced (respectively, consumed) by an activity has been matched to a corresponding consumer (respectively, producer) via producer/consumer commitments, we say that the activity has been *completely matched*. In a solution, a producer does not necessarily have to be completely matched since some of the inventory it produces may remain in storage at the end of the scheduling horizon. A consumer, however, must be completely matched.

The commitment scheme adopted here is that in a search state, S, where for inventory, I, there exists a consumer, c, that has not been completely matched, we can assert a producer/consumer commitment of the following form:

$$(p \rightarrow c, \text{min}(\text{amt-unmatched}(p,S), \text{amt-unmatched}(c,S)) \quad (5)$$

Where:

- p is a producer of I
- amt-unmatched(a, S) is the amount of inventory produced (or consumed) by a that has not been matched in state S
- amt-unmatched(a, S) > 0

If, through a complete retraction technique, we derive a dead-end, we post the commitment $\neg$ ($p \rightarrow c$, min(amt-unmatched(p, S), amt-unmatched(c, S)). This alternative commitment has the effect of removing from consideration another heuristic commitment between p and c until a state such that the minimum amount of unmatched inventory for p and c has changed. This branching scheme is similar to the "schedule versus postpone" branching scheme for start time assignments used in (Le Pape et al., 1994). Completeness of the branching scheme is proven in (Beck, 1999).

Producer/Consumer Interval Commitments

When all consumers are completely matched, an inventory minimum constraint cannot be violated (unless the problem is trivially unsatisfiable: the amount of inventory consumed is greater than the amount produced). All consumers have been paired with producers that supply sufficient inventory and, therefore, the inventory minimum constraint must be satisfied. A critical inventory maximum constraint, however, indicates that there is a non-zero probability that the inventory maximum constraint will be broken. To reduce this criticality, we constrain the time interval between the end of the producer and the start of the consumer. The intuition behind this commitment is that when a producer and consumer execute closer together in time, the average inventory level is decreased.

The actual commitment made is to constrain the interval between producer and consumer to be equal to or less than the middle value of the current domain of possible interval lengths. If such a commitment is retracted, we can post the opposite commitment: constraining the temporal interval to be greater than the middle value in the domain.

Start Time Commitments

Finally, it may be the case that all consumers are completely matched and all producer-consumer pairs are constrained to occur within a constant time of each other, but the search state is still neither a dead-end nor a solution. In such a state, it is necessary to assign start times to at least some of the activities.

Heuristic Commitment Techniques

Three classes of heuristic commitment techniques are used in our empirical investigations.

1. *Texture-based heuristics* – the defining characteristic of these techniques is the calculation of texture measurements on *both* inventory and resource constraints. The heuristic uses the texture information to dynamically decide which type of commitment to make.
2. *Non-texture-based inventory heuristics* – in this class, producer/consumer commitments are made by a non-texture-based heuristic before proceeding to a second heuristic commitment component to sequence the activities on each resource, followed (possibly) by a third heuristic to assign start times.
3. *Non-inventory heuristics* – this class is composed of heuristics which simply make commitments on the resources and use inventory propagation to maintain the inventory constraints.

It should be stressed that these techniques represent methods to form heuristic commitments. Other components of the scheduling algorithms (*i.e.*, propagators and retraction techniques) are orthogonal and complementary to the work reported here. In particular, the heuristic commitment techniques can be directly used in algorithms that also contain the inventory timetable mechanism of ILOG Scheduler and/or the cumulative constraint in CHIP.

Texture-based Heuristics

The texture measurements allow us to estimate the probability of breakage of resource maximum, inventory minimum, and inventory maximum constraints over time. The heuristic used in this paper, VarHeight, does the following:

1. Calculates texture measurements on all resource and inventory constraints.
2. Identifies the most critical constraint and time point, defined to be the constraint and time point with the highest probability of breakage (ties are broken arbitrarily).
3. Generates a commitment to reduce the criticality of the most critical constraint.

The heuristic commitment that is generated depends on which type of constraint is most critical. Therefore, the different types of commitments are interleaved throughout the scheduling process based on the texture information.

Resource Commitment

Given that resource, R^*, at time, t^*, has been identified as most critical, the VarHeight heuristic does the following:

1. Identifies the two activities, A and B, which rely most on R^* at t^* and that are not already connected by a path of temporal constraints.
2. Chooses the sequence ($A \rightarrow B$ or $B \rightarrow A$) based on sequencing heuristics. These sequencing heuristics are presented in (Beck et al., 1997b).

Minimum Inventory Commitment

When a minimum inventory constraint is found to be most critical, the heuristic identifies all consumers that can consume at or before the critical time point, and selects the one with the largest unmatched inventory. The activity is, heuristically, the most critical consumer.

Having the most critical consumer, c, we then examine all producers that can supply that consumer. We choose the producer with an earliest finish time less than or equal to the latest start time of the critical consumer which has some unmatched inventory and minimizes the value $lst_c - eft_p$. The justification for this heuristic is that it will tend to minimize inventory levels if the producer and consumer are able to execute close together in time. The commitment posted is a producer/consumer commitment between p and c as described in our branching scheme.

Maximum Inventory Commitment

When a maximum inventory constraint is critical, three types of commitments are made in the following order: producer/consumer, producer/consumer interval, and start time assignment. In each case, all commitments of one type are made before any commitments of the next. That is, no producer/consumer interval commitments will be made if it is still possible to make a producer/consumer commitment. Similarly, no start times will be assigned if it is still possible to make either of the other two commitments. Each of these commitment choices have their own sub-heuristics.

- Producer/Consumer Heuristic – when a maximum inventory constraint is critical, the heuristic identifies the producer, p, with the largest unmatched inventory that can produce at or before the critical time point. The consumers whose latest start times are greater than the earliest finish time of p are evaluated and, as with the inventory minimum heuristic, the consumer, c, that minimizes $lst_c - eft_p$ is chosen.
- Producer/Consumer Interval Heuristic – a producer-consumer pair, p and c, is heuristically selected such that the inventory transferred from p to c is the maximum of all producer-consumer pairs that meet the requirement that $eft_p \leq t^* \leq lst_c$, where t^* is the critical time point. As noted above, the actual commitment that is made is to limit the allowed time interval between the end of the producer and the start of the consumer.
- Start Time Assignment Heuristic – our start time assignment follows the “earliest or postpone” (*EorP*) branching scheme for start time assignments due to Le Pape et al. (Le Pape et al., 1994). All unassigned activities are ordered in ascending order of earliest start time with ties broken by ascending order of latest start time. A commitment results from selecting the first element on the list and assigning its earliest start time. If a dead-end is found, the activity is “postponed”: the activity can not take part in a start time commitment until propagation results in a new earliest start time for the activity.

Non-texture-based Inventory Heuristics

Without a technique to compare the criticality of inventory and resource constraints, there does not appear to be a principled way to interleave the types of heuristic commitments. The commitments are therefore made in the following order: producer/consumer, resource, and start time. As with the inventory maximum heuristic, all commitments of one type are made before any commitments of the next. Different heuristics are used for each commitment:

- Producer/Consumer Commitment Heuristic – the non-texture-based producer/consumer heuristic selects consumers in descending order of their amount of unmatched consumption. An upstream producer is identified based on the minimization of $lst_c - eft_p$. We refer to this heuristic as *GreedyInv.*
- Resource Commitment Heuristics – any of the techniques used for job shop scheduling can be used for the resource sequencing heuristic. In our experiments, we use SumHeight and CBASlack. The *SumHeight* heuristic (Beck et al., 1997b) uses the contention texture measurement to estimate the criticality of all resources and then sequences the two most critical activities on the most critical resource using the same sequencing heuristics used by VarHeight. SumHeight is used here, rather than the more general VarHeight, because the former has been shown to outperform the latter on job shop scheduling problems (Beck, 1999). The Constraint-Based Analysis Slack (*CBASlack*) heuristic (Cheng and Smith, 1997) identifies the unsequenced activity pair with the smallest biased slack. This pair is then sequenced to preserve the most slack. A resource commitment heuristic does not necessarily have to be defined. In the case of a null resource commitment heuristic, the start-time assignment heuristic is used after all inventory commitments are made.
- Start Time Assignment Heuristics – it is possible to reach a search state such that all consumers are completely matched and all activities on the resources are sequenced,

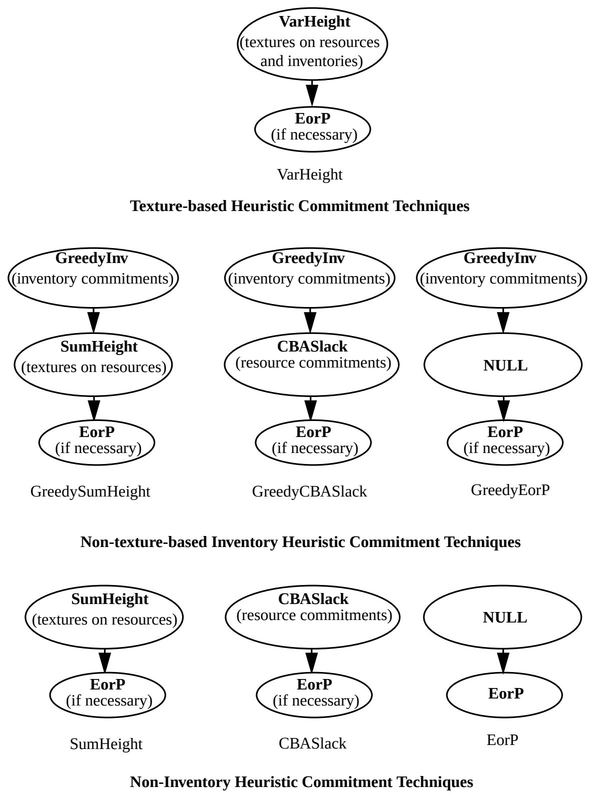

Figure 3. Schematic Representation of the Three Classes of Heuristic Commitment Technique.

yet the termination criteria on one or more inventories are not met. In such a search state, start time assignments are necessary. The start time assignment heuristic used is the EorP heuristic described above.

Scheduling Without Inventory Heuristics

To provide a basis of comparison for our inventory heuristics against heuristics typical of those in the literature, we use the resource sequencing heuristics and start time assignment heuristics discussed in the previous section. We run one of our resource heuristics (SumHeight, CBASlack, or NULL) followed, if necessary, by the EorP start-time assignment heuristic.

Summary

The three classes of heuristic commitment technique are shown in Figure 3. There is a single instance of a texture-based heuristic commitment technique (VarHeight) and three instances each of the other two classes. The name below each technique corresponds to the scheduling policy using that technique.

Empirical Evaluation

The primary goal in our experiments is to understand the performance differences among the heuristic commitment techniques. To this end, we undertake two experiments. The first uses one-stage problems to investigate heuristic performance as the number of inventory consumptions is increased. The second experiment turns to two-stage problems and manipulates the overall scheduling horizon.

The heuristic commitment techniques are the sole difference among the scheduling policies. All algorithms use chronological backtracking and the following propagators:

- temporal propagation (Lhomme, 1993)
- constraint-based analysis (Cheng and Smith, 1997)
- edge-finding exclusion (Nuijten, 1994)
- edge-finding not-first/not-last (Nuijten, 1994)
- inventory bound propagation (Beck, 1999)
- producer/consumer propagation (Beck, 1999)

The final two propagation techniques have been shown to significantly improve the overall performance of scheduling policies on inventory scheduling problems (Beck, 1999).

The use of chronological backtracking as opposed to a discrepancy-based retraction technique (Harvey and Ginsberg, 1995; Walsh, 1997) is motivated by our focus on the heuristics and the relative lack of previous work on scheduling with inventory. Future work will explore the effect of such techniques.

The Reporting of Time-outs

Our experiments are run with a bound on the CPU time. Each algorithm must either find a schedule or prove that no schedule exists for a problem instance. If an algorithm is unable to do so within the CPU time limit (in all our experiments the limit is 20 minutes), a time-out is recorded.

The primary reason for reporting time-out results is that it allows us to use problem sets that contain both soluble and over-constrained problems. The phase transition work in combinatorial problems such as SAT and CSP (Gent and Walsh, 1994; Gent et al., 1996) demonstrates that the hardest problem instances are found in locations of the problem space where approximately half of the problems are over-constrained. While the space of scheduling problems is not as well-understood as SAT or CSP in terms of phase transition phenomena (Beck and Jackson, 1997), we want to take advantage of this insight in order to generate challenging problem instances. We construct our problem sets so that as the independent variable changes, the problem instances move from an over-constrained area in the problem space to an under-constrained area. In the former area, proofs of insolubility can often be easily found while in the latter area, solutions can be easily found. It is in the middle range of problems where we expect to find the most difficult problems.

Unless otherwise noted, statistical significance for each performance measure is evaluated with the boot-strap paired-t test (Cohen, 1995) with $p \leq 0.0001$.

Experiment 1: One-Stage Problems

The problems used in this experiment are 10×10 one-stage inventory scheduling problems. The makespan factor is held constant at 1.2 and the primary independent variable is c, the number of consumptions per job. Rather than directly manipulating c, we manipulate $p = c/nm$, the proportion of

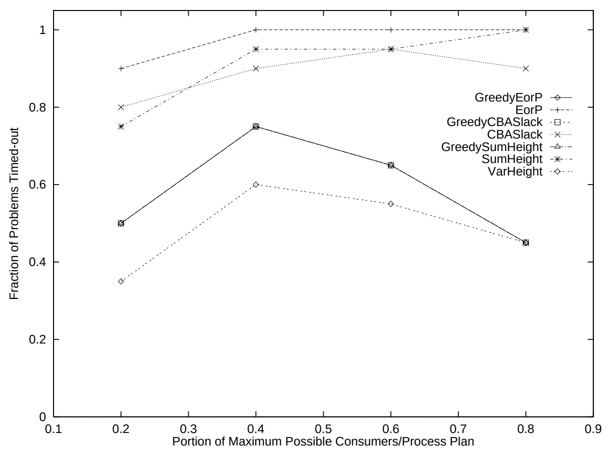

Figure 4. The Fraction of One-Stage Problems Timed-out for Each Problem Set and Algorithm.

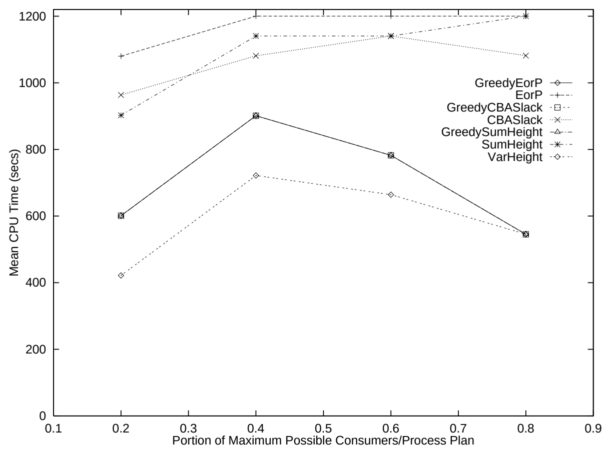

Figure 5. The Mean CPU Time in Seconds for Each One-Stage Problem Set and Algorithm.

the maximum possible consumers for each job. For our experiments, *p* varies from 0.2 to 0.8 in steps of 0.2. For each value of *p*, twenty problems are generated. At low values of *p*, there are few consuming activities and therefore the combination of temporal and inventory constraints should be relatively easy to satisfy: there are few activities that must be scheduled to consume from the predefined supply events. In contrast, at high values of *p*, there are many such consumers and, therefore, we expect problems to be more difficult to solve and, in many cases, insoluble. The makespan factor was chosen based on the results of job shop problems that showed that most of the random job shop problems are soluble at such a factor (Beck, 1999).

The timed-out results are presented in Figure 4 while the mean CPU time results are in Figure 5. Similar performance among a number of the algorithms obscures some of the results. In Figure 4, all the algorithms using the non-texture-based inventory heuristics (*i.e.*, GreedySumHeight, GreedyCBASlack, and GreedyEorP) achieve the same results and coincide on the plot beginning at (0.2, 0.5). In

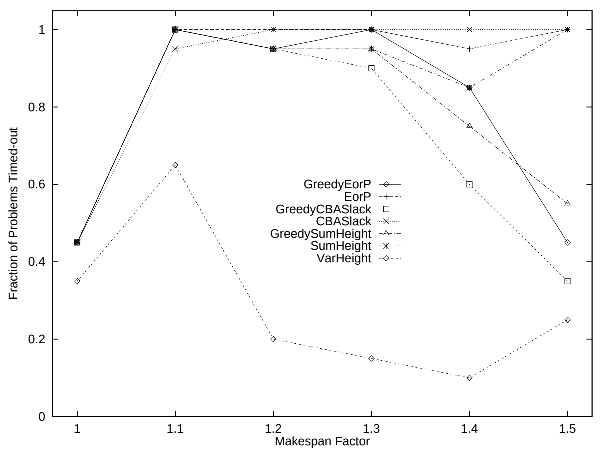

Figure 6. The Fraction of Two-Stage Problems Timed-out for Each Problem Set and Algorithm.

Figure 5, the non-texture inventory heuristics achieve identical performance within a few hundredths of a second and coincide on the plot beginning at (0.2, 601.0).

Statistically, VarHeight times-out on significantly fewer problems than SumHeight, CBASlack, and EorP while showing no significant difference when compared to the algorithms using non-texture-based inventory heuristics. In comparing the corresponding pairs of algorithms, (*e.g.*, GreedySumHeight with SumHeight) we see that each non-texture-based inventory heuristic times-out on significantly fewer problems than its non-inventory heuristic counterpart ($p \leq 0.0005$).

Turning to the CPU time results, the statistical analysis reveals that VarHeight incurs significantly less CPU time than each of the algorithms using non-inventory heuristics ($p \leq 0.0005$). Examining the corresponding pairs of non-texture heuristics versus non-inventory heuristics, we observe that each algorithm using a non-texture inventory heuristic incurred significantly less CPU time than the corresponding one using non-inventory heuristics. There are no other significant differences. In particular, there are no significant differences between VarHeight and any of the non-texture-based inventory heuristics.

Experiment 2: Two-Stage Problems

Experiment 2 uses two-stage problems, each formed from two 10$\times$10 one-stage problems. The same algorithms as Experiment 1 are used. The proportion of possible consumers, *p*, is held constant at 0.4. This value was chosen, based on Experiment 1, in order to generate relatively difficult problems. The independent variable is the makespan factor which is varied from 1.0 to 1.5 in steps of 0.1. For each makespan factor, twenty problems are generated.

The proportion of problems in each problem set for which the algorithms time-out is shown in Figure 6. In general, the algorithms using non-inventory heuristics do poorly on all problem sets from makespan factor 1.1 to makespan factor 1.5. The algorithms using non-texture-based inventory heuristics perform poorly on problem sets

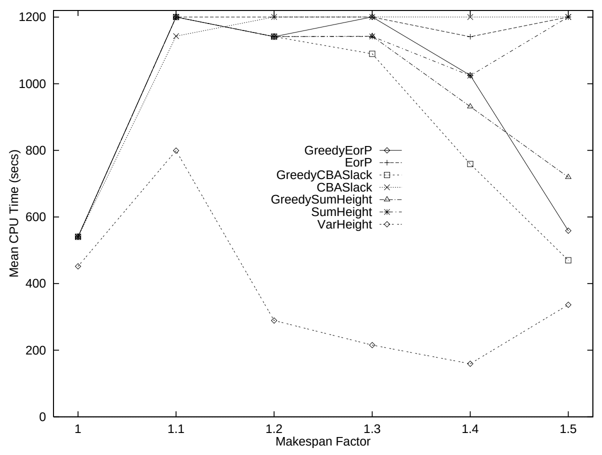

Figure 7. The Mean CPU Time in Seconds for Each Two-Stage Problem Set and Algorithm.

with low makespan factors (1.1 and 1.2), but improve as the makespan factor increases. The mean CPU time for each algorithm is shown in Figure 7. VarHeight incurs lower mean CPU time than all other algorithms across all problem sets. The other algorithms perform similarly on the problem sets with low makespan factor, but diverge at higher makespan factors. As above, the algorithms with non-texture-based heuristics appear to improve (relative to the non-inventory heuristic algorithms) with the higher makespan factors.

Statistical analysis reveals that VarHeight times-out on significantly fewer problems and incurs significantly less mean CPU time than all other algorithms. In comparing the non-texture-based heuristics with their non-inventory heuristic counterparts, we see that the former significantly outperform the latter ($p \leq 0.005$) in terms of the number of problems timed-out and the mean CPU time.

Discussion

The experiments show that the use of texture-based heuristics results in as good as or better performance than the use of non-texture-based inventory heuristics and non-inventory heuristics. In particular, when the inventory relationships among activities are made more complex, the algorithm using a texture-based heuristic significantly outperforms all other algorithms on all problem sets tested.

The motivation for a measure of criticality that can be applied to a wide range of constraints is that, at each search state, we want to be able to identify the most critical constraint, regardless of type, and focus our heuristics on reducing the criticality. The contributions of the dynamic focus ability are particularly evident in Experiment 2. As we argue below, as the makespan factor increases the relative criticality of the resource and inventory constraints changes. Throughout the change, VarHeight outperforms all other algorithms. We interpret these results as support for our approach to scheduling via texture measurement-based heuristic search. The ability to not only identify the constraints that are more critical at the beginning of the problem but also in every search state allows the texture-based heuristics to achieve high-quality search on problems with varying characteristics.

The experiments demonstrate that taking inventory into account when making heuristic decisions, even with a greedy heuristic, leads to better overall search performance than when the heuristics focus solely on resource constraints and allow propagators to maintain the inventory constraints. This observation would seem to be obvious, given problems that are strongly characterized by inventory constraints. Nonetheless, the few scheduling strategies in the literature that address inventory problems tend to account for inventory via inventory propagators alone.

In Experiment 2, for low values of the makespan factor, there is little slack on each resource, and therefore the criticality of the resource constraints is high. When the makespan is larger, however, the resource constraints are easier to satisfy and therefore the inventory constraints become, relatively, more critical. On this basis, then, we would expect that a heuristic that focuses on resource constraints will perform well for problem sets with low makespan factors while at larger factors heuristics that put more effort toward the inventory constraints will be superior. These are the results we see: the non-inventory heuristics perform well at makespan factors 1.0 and 1.1 and very poorly on the problem sets with makespan factors 1.4 and 1.5. The non-texture-based inventory heuristics, in contrast, improve with higher makespan factors.

In many problem sets, the non-texture-based inventory heuristics achieved identical performance. Analysis reveals that each of the algorithms is dependent on the GreedyInv heuristic: either GreedyInv found a set of producer/consumer commitments that could be extended to an overall solution with no further backtracking or it was unable to find a set of consistent producer/consumer commitments.

Conclusions

The central thesis of this paper is that an understanding of the structure of a problem leads to high-quality heuristic problem solving performance in constraint-directed scheduling. Our methods for gaining an understanding of problem structure focus on *texture measurements*: algorithms that implement dynamic analyses of each search state. Texture measurements distill structural information from the constraint graph which is then used as a basis for heuristic decision making.

We formulated a texture measurement to estimate the probability of breakage of resource and inventory constraints. The ability to estimate the criticality of both resource and inventory constraints allows heuristics to dynamically and opportunistically reason about the most critical constraint in a problem state, independent of whether the constraint is a resource or inventory constraint.

It was empirically demonstrated that the ability to dynamically focus on the most critical constraint in a problem state leads to significantly better overall heuristic search performance. When the experimental problems are manipulated to have more complex producer/consumer relationships, it was shown that the dynamic focusing abilities arising from an understanding of the problem structure is particularly important to successful heuristic search. These results support the problem structure hypothesis that we set out to investigate in this paper. The identification and exploitation of the problem structure revealed by an estimate of the constraint criticality leads to better overall heuristic problem solving performance.

Finally, it should be noted that the techniques presented fit wholly within the constraint-directed scheduling approach. Therefore, a broader contribution of this paper is the demonstration of the flexibility and extensibility of the constraint-directed approach to scheduling. Indeed, the ability to represent and reason about the myriad of constraints relevant to real-world scheduling problems was one of the original motivations for applying constraint technology to scheduling (Fox, 1983).

Acknowledgements

The research reported in this paper was performed while the author was a Ph.D. student in the Department of Computer Science, University of Toronto. It was funded, in part, by the Natural Sciences Engineering and Research Council, IRIS Research Network, Manufacturing Research Corporation of Ontario, Baan Limited, and Digital Equipment of Canada.

Thanks to Andrew Davenport, Angela Glover, Edward Sitarski, Jean-Paul Watson, and Mark Fox for discussion of and comments on previous versions of this paper.

References

Beck, J. C. (1999). *Texture measurements as a basis for heuristic commitment techniques in constraint-directed scheduling*. PhD thesis, University of Toronto.

Beck, J. C., Davenport, A. J., Davis, E. D., and Fox, M. S. (1998). The ODO project: Toward a unified basis for constraint-directed scheduling. *Journal of Scheduling*, 1(2):89–125.

Beck, J. C., Davenport, A. J., Sitarski, E. M., and Fox, M. S. (1997a). Beyond contention: extending texture-based scheduling heuristics. In *Proceedings of Fourteenth National Conference on Artificial Intelligence (AAAI-97)*. AAAI Press, Menlo Park, California.

Beck, J. C., Davenport, A. J., Sitarski, E. M., and Fox, M. S. (1997b). Texture-based heuristics for scheduling revisited. In *Proceedings of Fourteenth National Conference on Artificial Intelligence (AAAI-97)*. AAAI Press, Menlo Park, California.

Beck, J. C. and Jackson, K. (1997). Constrainedness and the phase transition in job shop scheduling. Technical report, School of Computing Science, Simon Fraser University.

Blazewicz, J., Domschke, W., and Pesch, E. (1996). The job shop scheduling problem: Conventional and new solution techniques. *European Journal of Operational Research*, 93(1):1–33.

Carlier, J. and Pinson, E. (1994). Adjustment of heads and tails for the job-shop problem. *European Journal of Operational Research*, 78:146–161.

Cheng, C. C. and Smith, S. F. (1997). Applying constraint satisfaction techniques to job shop scheduling. *Annals of Operations Research, Special Volume on Scheduling: Theory and Practice*, 70:327–378.

Cohen, P. R. (1995). *Empirical Methods for Artificial Intelligence*. The MIT Press, Cambridge, Mass.

Fox, M. S. (1983). *Constraint-Directed Search: A Case Study of Job-Shop Scheduling*. PhD thesis, Carnegie Mellon University, Intelligent Systems Laboratory, The Robotics Institute, Pittsburgh, PA. CMU-RI-TR-85-7.

Fox, M. S., Sadeh, N., and Baykan, C. (1989). Constrained heuristic search. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 309–316.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York.

Gent, I. P., MacIntyre, E., Prosser, P., and Walsh, T. (1996). The constrainedness of search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, volume 1, pages 246–252.

Gent, I. P. and Walsh, T. (1994). The hardest random SAT problems. In Nebel, B. and Dreschler-Fischer, L., editors, *Proceedings of KI-94: Advances in Artificial Intelligence. 18th German Annual Conference on Artificial Intelligence*, pages 355–366. Springer-Verlag.

Harvey, W. D. and Ginsberg, M. L. (1995). Limited discrepancy search. In *Proceedings of the Fourteenth International Joint Conference onf Artificial Intelligence (IJCAI-95)*, pages 607–613.

Le Pape, C. (1994a). Implementation of resource constraints in ILOG Schedule: A library for the development of constraint-based scheduling systems. *Intelligent Systems Engineering*, 3(2):55–66.

Le Pape, C. (1994b). Using a constraint-based scheduling library to solve a specific scheduling problem. In *Proceedings of the AAAI-SIGMAN Workshop on Artificial Intelligence Approaches to Modelling and Scheduling Manufacturing Processes*.

Le Pape, C., Couronné, P., Vergamini, D., and Gosselin, V. (1994). Time-versus-capacity compromises in project scheduling. In *Proceedings of the Thirteenth Workshop of the UK Planning Special Interest Group*.

Lhomme, O. (1993). Consistency techniques for numeric CSPs. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, volume 1, pages 232–238.

Muscettola, N. (1992). Scheduling by iterative partition of bottleneck conflicts. Technical Report CMU-RI-TR-92-05, The Robotics Institute, Carnegie Mellon University.

Nuijten, W. P. M. (1994). *Time and resource constrained scheduling: a constraint satisfaction approach*. PhD thesis, Department of Mathematics and Computing Science, Eindhoven University of Technology.

Sadeh, N. (1991). *Lookahead techniques for micro-opportunistic job-shop scheduling*. PhD thesis, Carnegie-Mellon University. CMU-CS-91-102.

Saks, V. (1992). Distribution planner overview. Technical report, Carnegie Group, Inc., Pittsburgh, PA, 15222.

Simon, H. A. (1973). The structure of ill-structured problems. *Artificial Intelligence*, 4:181–200.

Simonis, H. and Cornelissens, T. (1995). Modelling producer/consumer constraints. In Montanari, U. and Rossi, F., editors, *Proceedings of the First International Conference on Principles and Practice of Constraint Programming (CP95)*, pages 449–462. Springer-Verlag.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64:278–285.

Walsh, T. (1997). Depth-bounded discrepancy search. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*.