ELSEVIER

Artificial Intelligence 121 (2000) 211–250

Artificial Intelligence

www.elsevier.com/locate/artint

Constraint-directed techniques for scheduling alternative activities

J. Christopher Beck [a,*], Mark S. Fox [b]

[a] *ILOG, SA, 9 rue de Verdun, 94253 Gentilly Cedex, France*

[b] *Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Ontario, Canada M5S 3G9*

Received 12 October 1999

Abstract

In this paper, we expand the scope of constraint-directed scheduling techniques to deal with the case where the scheduling problem includes alternative activities. That is, not only does the scheduling problem consist of determining when an activity is to execute, but also determining which set of alternative activities is to execute at all. Such problems encompass both alternative resource problems and alternative process plan problems. We formulate a constraint-based representation of alternative activities to model problems containing such choices. We then extend existing constraint-directed scheduling heuristic commitment techniques and propagators to reason directly about the fact that an activity does not necessarily have to exist in a final schedule. Experimental results show that an algorithm using a novel texture-based heuristic commitment technique together with extended edge-finding propagators achieves the best overall performance of the techniques tested.

Keywords: Scheduling; Heuristic search; Constraints; Problem structure; Alternative activities

1. Introduction

The scheduling problems addressed in the constraint-directed scheduling literature typically have a static activity definition: each activity *must* be scheduled on its specified resource(s). It is common, however, in real-world scheduling problems to have a wider space of choices. Typically, in a manufacturing setting, there are choices to be made in scheduling among alternative resources on which to run an activity or among

* Corresponding author.

E-mail addresses: cbeck@ilog.fr (J.C. Beck), msf@eil.utoronto.ca (M.S. Fox).

0004-3702/00/$ – see front matter
PII: S0004-3702(00)00035-7

alternative process plans (also called "routings") of an order through a factory. While this characteristic is ubiquitous in industry, there has been little examination of such alternatives in the constraint-directed scheduling literature.

The central contribution of this paper is the expansion of scope of scheduling problems that can be addressed by constraint-directed scheduling techniques. This expansion is done by integrating the representation of activity alternatives into the constraint graph representation of the scheduling problems. We then extend heuristic techniques and sophisticated propagators to account for the richer representation.

There are two primary motivations for the work in this paper. The first is the expansion of the scope of constraint-directed scheduling techniques. While there have been considerable advances over the past few years in the scope and difficulty of scheduling problems that can be successfully addressed with constraint technology [6], there are still a number of problem characteristics that have been examined only partially or not at all in the literature. These characteristics are nonetheless important from the practical perspective of modeling and solving problems as they exist *in situ*. Choosing resources on which to schedule an activity as well as amongst different sets of activities with which to achieve the same goals, are two such characteristics. The second motivation for the work in this paper is the investigation of the problem structure hypothesis due to Simon [41]. The *problem structure hypothesis* states that as a problem becomes more complex, understanding of its structure becomes increasingly important for successful heuristic search. While the problem structure hypothesis has been pursued in the context of scheduling by Fox et al. [4,5,7,8,10,21,22,36,37], the inclusion of activity alternatives results in an additional dimension of complexity. Therefore, scheduling with alternative activities is a prime application for the investigation of the hypothesis.

1.1. Plan of paper

The plan of this paper is as follows: in the following section, we provide the necessary background for the work in this paper. In Section 3, the representation of alternatives is presented, while Section 4 and Section 5, respectively, examine propagators and heuristic commitment techniques for alternative activity problems. Our empirical evaluations appear in Section 6 through Section 8 with the discussion of our results appearing in Section 9. In Section 10, we briefly note future work stemming from this paper before concluding in Section 11.

2. Background

Before presenting the extensions to constraint-directed scheduling representation and techniques to incorporate the activity alternatives, we define the problems of alternative resources and alternative process plans, and discuss previous work that has examined these two characteristics. In addition, we introduce two components of constraint-directed scheduling algorithms that have been previously applied to non-alternative activity problems: texture-based heuristic commitment techniques and propagators.

Table 1
Notation

Symbol	Description
ST_i	a variable representing the start time of A_i
STD_i	the discrete domain of possible values for ST_i
est_i	earliest start time of A_i
lst_i	latest start time of A_i
dur_i	duration of A_i
eft_i	earliest finish time of A_i
lft_i	latest finish time of A_i

2.1. Notation

For an activity, A_i we use the notation in Table 1 through the balance of this paper. We will omit the subscript unless there is the possibility of ambiguity.

2.2. Problem definition

In a basic scheduling problem (such as the job shop scheduling problem [23]) each activity must be assigned a time to execute on a predefined resource. As each activity must be executed and the resource is already specified, the only component of the problem that must be decided is when each activity will execute.

In an *alternative resource scheduling problem*, rather than having a unique resource that it must execute on, each activity may have a set of alternative resources and may execute on any member of the alternative set. Depending on the resource chosen, other characteristics of the activity (e.g., duration) may also change. A solution to the problem must specify which resource the activity executes on as well as when it executes.

In an *alternative process plan scheduling problem*, the choices are expanded to encompass multiple sets of activities. Fig. 1 displays four alternative process plans

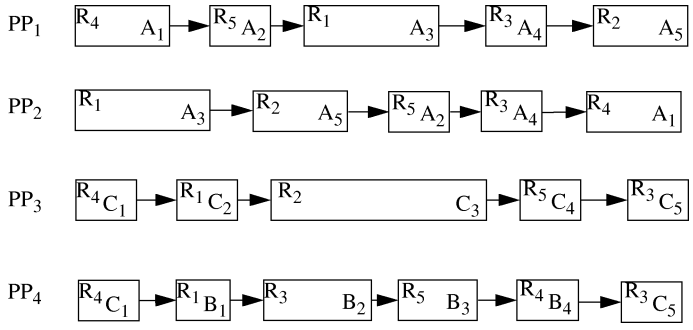


Fig. 1. Four alternative process plans.

($PP_1, \ldots, PP_4$). The label in the upper-left corner of each activity represents the activity's resource requirement while the lower-right label is the identifier of the activity. Thus, activities with the same identifier (e.g., A_3 in PP_1 and A_3 in PP_2) are the same. The first two process plans, PP_1 and PP_2, are simply different orderings of the same activities. The third process plan, PP_3, is a completely different recipe while the fourth, PP_4, is a variation on the third: the first and last activities are identical, but the middle ones are different. Only one of the alternative process plans is to be executed. Therefore, the scheduling problem not only consists of deciding *when* to execute the activities but *which* of the alternative process plans will be executed at all.

2.3. Previous work

2.3.1. Alternative resources

In its most basic form, the alternative resource problem can be represented with the use of multi-capacity resources [31]. A resource of capacity k can be used to model a resource set of k identical unary capacity resources and the activities can simply be scheduled on the multi-capacity resource without representing the unary capacity resources. Significant representational savings can be gained from this type of representation. Unfortunately, it is unclear if elaborations of the resource model, such as requiring changeover activities [11], can be represented in this way.

ISIS [21] provided representation for alternative resources within an order-based *Incremental Decomposition and Incremental Scheduling* (IDIS) [27] approach. All the activities for one order are assigned start times and resource assignments before the next order is selected for scheduling. A similar order-based decomposition is used in the Dynamic Scheduling System (DSS) [26] in the context of on-line scheduling. Based on a blackboard system and stochastic order arrival, DSS instantiates process plans with resource alternatives that include preference information. Based on analyses of the existing schedule and the preferences, resource reservations are created for each of the newly instantiated activities. While the scheduling process typically assigns activities to resources in one order before moving to the next, in more challenging scheduling problems, DSS may cancel previous reservations in order to search for a global solution among the currently instantiated activities.

In a constraint programming language, such as ILOG Solver and Scheduler [1], alternative resources can be modeled with one boolean variable for each resource-choice of an activity. An activity uses a resource only if the corresponding demand variable is TRUE. By using a constraint stating that only one of the demand variables can be TRUE, the alternative resource requirements can be modeled and demand variables can be incorporated into solution techniques.

A combination of the multi-capacity representation and the boolean representation can be done by using multiple representations of the same resource requirements [31]. Multi-capacity resources are used to represent resource sets while the unary capacity resources are directly represented. A multi-capacity version of the edge-finding exclusion propagator has been implemented and used on the multi-capacity resources at the same time as edge-finding and other propagators are used on the unary capacity resources. Extensions of these techniques allow activities to have overlapping resource sets and allow

the duration of an activity to depend on the assigned resource alternative. Building on the use of multiple representations, Davenport et al. [19] formulate a number of heuristic commitment techniques for the alternative resource problems. Again, both the multi-capacity resource and unary capacity resources are explicitly represented. The central algorithm for scheduling has two phases: in the first, all the activities are assigned to one of their resource alternatives, and in the second phase the activities on the unary capacity resources are sequenced using job shop scheduling techniques.

2.3.2. Alternative process plans

Kott and Saks [27] introduce a spectrum of approaches to alternative process plan scheduling. At one extreme, the *Multiple Alternative Decomposition* (MAD) approach, the alternatives are fully represented and integrated in the scheduling process. The other extreme is the *Complete Decomposition Prior to Scheduling* (CDPS) approach where all alternatives are decided in a pre-scheduling phase of the search. An intermediate approach along this spectrum is the *Incremental Decomposition and Incremental Scheduling* (IDIS) approach where the search through the alternatives and the scheduling is interleaved.

The order-based decomposition approach of ISIS discussed above can be applied to problems containing alternative process plans [21]. In addition to alternative resources, the order representation in ISIS allows alternative routings of a single order. After such an order is chosen for scheduling, the resource allocation heuristics are used to choose a single routing and resource reservations for all activities along the chosen routing.

The KBLPS scheduling system [14,39] builds on the order-based decomposition of ISIS by the direct representation of alternative process plans and the incorporation of alternative information into texture-based heuristics. In KBLPS, a probabilistic representation of all activities similar to that of Sadeh [36,37] was used. The innovation was to recognize that the probability that a particular routing is chosen depends on the number of alternative routings that are available: the individual resource demand for each activity in a routing was therefore biased to represent the likelihood that the routing itself was chosen. The resource reservation heuristics, therefore, have deeper information on which to base their commitments. Kott and Saks [27] extend the KBLPS work by embedding alternative process plan scheduling in the context of combined planning and scheduling. Viewing alternative process plans as the result of goal decomposition, the authors examine the scheduling of multiple, alternative decompositions. Using domain specific rules, a set of alternative process plans are constructed for each goal to create the scheduling problem. Then, based on the contention and reliance texture measurements, the activity most reliant on the most contended-for resource is selected. As in KBLPS, the individual demand of each activity is biased by its probability of execution given the alternative process plans that are available to achieve the same goal. The critical activity is used to identify the critical goal, and a single process plan is selected to achieve that goal based on a combination of domain specific rules and the reliance of the critical activity. All the activities in the chosen process plan are scheduled.

A higher-level approach to alternative process plan scheduling can be seen in work that integrates the process planning task with scheduling. Systems such as Design-to-Criteria [43,44] and IP3S [38] dynamically create process plans according to a number of criteria. The Design-to-Criteria work takes into account a combination of uncertainty,

cost, and time-to-completion in exploring the space of plans with which a set of goals can be achieved. Based on the high-level evaluation, a set of activities without alternatives is selected and scheduled. Depending on the quality of the resulting schedule, decisions from the planning phase may be revisited in order to achieve a better overall schedule. Similarly, IP3S uses a black-board architecture to integrate process planning and scheduling. IP3S achieves this integration by taking into account the resource contention when formulating process plans. As with Design-to-Criteria, non-alternative process plans are generated and scheduled; however, based on the quality of the schedule, process plans may be generated specifically to route process plans away from highly contended-for resources. In both systems, however, the alternative process plans are represented only in the planning phase: no representation of alternatives is present during the actual scheduling.

Le Pape [28] extends the activity representation of ILOG Scheduler to allow an activity's duration to be 0 to represent that the activity does not execute. In an application that represents activities with varying durations, all activities are initially present in the problem and the heuristic commitment technique has the choice of setting an activity's duration to 0, setting it to some lower-bound, and/or assigning a start time. This technique allows the representation alternative activities; however, no information is given on what, if any, propagation is done to reason about activity dependencies.

2.3.3. Discussion

We draw two general themes from the literature on activity alternatives. The first is the constraint programming approach represented by ILOG Scheduler where an alternative is represented by a full variable in the constraint representation of the problem. The second theme is the heuristic approach present in KBLPS and [27]: resource and start time reservations are made with texture-based heuristics which take into account the fact that alternative choices change the individual demand an activity has for a resource. These two themes are consistent with the work on scheduling non-alternative activities where propagators and sophisticated heuristic techniques are two key solution techniques.

Based on the foundation provided by the literature, our approach to scheduling with alternative activities is as follows:

(1) Represent the fact that an activity has alternatives and therefore may not actually execute in the constraint graph.
(2) Formulate and/or adapt texture-based heuristics so that they take into account the existence of alternatives and directly reason about them in generating heuristic commitments.
(3) Formulate and/or adapt propagators so that they too can take into account the existence of alternatives yet still derive implied commitments that can be added to the constraint graph.

3. Representation of alternative activities

In order to represent the fact that an activity present in the problem definition may not be present in a final schedule, we introduce the notion of an activity's *probability of existence* (PEX). Informally, PEX is defined to be the probability that an activity will be present in the

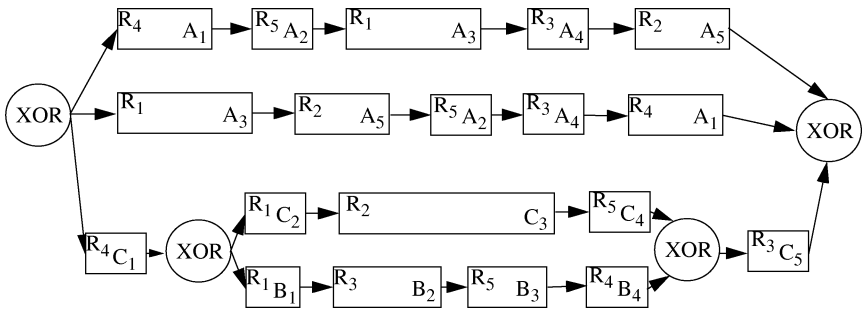


Fig. 2. Modification of the temporal network to directly model the alternatives implicit in Fig. 1.

final solution (assuming such a solution exists). The PEX of each activity is represented by a PEX variable, which is a standard domain variable with an initial domain of $\{0, 1\}$. In each search state, we calculate the expected value [1] of the PEX variable and use it to represent the activity's probability of existence. In addition, the temporal network is modified so that the expected PEX values and temporal values are propagated among activities. One of the key components of this extension is a modification of the temporal graph to explicitly represent alternative activities.

A general idea of the representational approach can be seen in Fig. 2. The XorNodes in the constraint graph directly represent the alternative process plans (and sub-process plans) from Fig. 1. We discuss the addition of XorNodes and PEX variables to the constraint graph in this section.

3.1. Extending the temporal graph

To represent activity alternatives directly in the temporal graph, we introduce AndNodes and XorNodes in additional to the regular Activity nodes. The temporal graph has multiple instances of AndNodes, XorNodes, and Activities connected via temporal constraints.

3.1.1. AndNode

The AndNode has start time and end time temporal variables just as an Activity does. The duration of an AndNode is 0. There are two semantic components in the AndNode corresponding to its functionality in temporal and PEX propagation. From a PEX perspective, all the TemporalNodes linked to an AndNode exist in a solution if the AndNode exists. Similarly, if one of the non-XorNodes directly linked to an AndNode exists in a solution, then the AndNode must exist. In terms of expected PEX values, therefore, all non-XorNodes directly connected to an AndNode must have the same expected PEX value as the AndNode itself. Since all the nodes an AndNode is connected to are either going to be present in a solution, or all the nodes, including the AndNode itself, are removed, it is valid to perform standard temporal propagation.

[1] We use the phrases "expected value" and "expected PEX value" informally here, by analogy to the usual statistical definition of the expected value of a random variable. A formal justification for the use of the phrases in their statistical sense is beyond the scope of this document.

3.1.2. XorNode

Like the AndNode, the XorNode has a duration of 0, and start and end time temporal variables. As the name indicates, the logical semantics of a XorNode are such that if the XorNode itself is present in the final solution then one and only one of the nodes directly connected upstream (respectively, downstream) to the XorNode can also be present. Also, if a node directly connected upstream (or downstream) to a XorNode exists in a solution, so must the XorNode. From a PEX propagation perspective, these semantics mean that, in a consistent network, the expected PEX value of the XorNode is equal to the sum of the expected PEX values from the nodes directly connected upstream and the XorNode expected PEX value must also be equal to the sum of the PEX values from the nodes directly connected downstream.

Fig. 3 represents a small temporal graph with XorNodes, X_1, X_2, and X_3. Assuming that the expected PEX value of X_1 and X_3 are both 0.5, the rest of the temporal nodes have the expected PEX values shown (above or below each node).

From a temporal perspective, the fact that only one of the upstream and one of the downstream links are to be present in a solution means that temporal propagation is different from that of the AndNode. A XorNode must end before the *maximum* latest start time of all the nodes directly connected downstream while it must start after the *minimum* earliest end time of all activities connected upstream. Assuming that the nodes between X_1 and X_3 must be scheduled within a scheduling horizon of $[0, 100]$, Table 2 displays the start and end time windows of each node. Note that for the temporal propagation of the AndNodes, they are treated simply as activities with zero duration. The XorNodes also have zero duration, but a more complicated treatment within temporal propagation due to their semantics. We revisit the issue of PEX and temporal propagation in Section 4.

3.1.3. Illegal temporal networks

We have extended our temporal network representation to allow definition of AndNodes and XorNodes. Not all temporal networks that are expressible are valid networks, however. The basic requirement for legality is that for any XorNode, X, in the graph with k upstream and l downstream links ($k, l > 1$), there must be:

- a corresponding XorNode, X^-, upstream of X with k downstream links, and
- a corresponding XorNode, X^+, downstream of X with l upstream links.

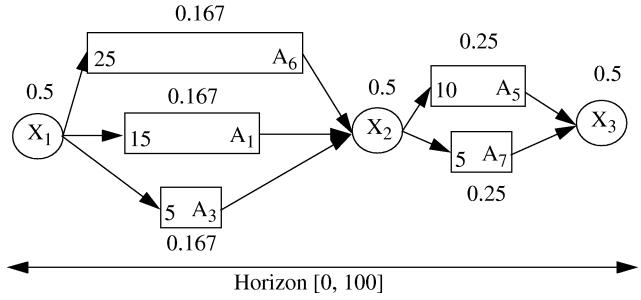


Fig. 3. A temporal graph with XorNodes.

Table 2
The time windows for the activities in Fig. 3

Node	Start time domain	End time domain
X_1	[0 90]	[0 90]
A_6	[0 70]	[25 95]
A_1	[0 80]	[15 95]
A_3	[0 90]	[5 95]
X_2	[5 95]	[5 95]
A_5	[5 90]	[15 100]
A_7	[5 95]	[10 100]
X_3	[10 100]	[10 100]

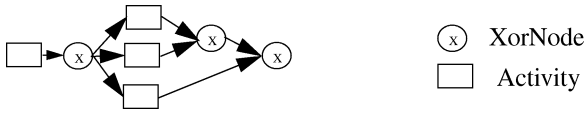


Fig. 4. An illegal temporal network.

For example, the temporal network in Fig. 4 clearly breaks our basic requirement for corresponding linkage of XorNodes. The first XorNode has 3 downstream links while there are no other XorNodes in the network with 3 upstream links. Intuitively, the reason that this network is illegal is an ambiguity for expected PEX values. What should the expected PEX values be for the middle three activities? Starting from the upstream activity, we would assign a PEX of 1 to it and to the first XorNode. Following the PEX propagation rules for XorNodes we would then calculate that each of the middle three activities have an expected PEX value of 0.33. Consider, however, starting from the last node and doing the PEX propagation upstream. The final XorNode will have an expected PEX value of 1. The two nodes directly connected to the final XorNode (i.e., the penultimate XorNode and the bottom activity) would then have an expected PEX value of 0.5 each and then propagating from the penultimate XorNode, the top two middle activities would have an expected PEX value of 0.25 each. The ambiguity over the expected values is the reason that network B is illegal and therefore the reason for the "matching links" requirement for XorNodes in a legal temporal network.

3.2. Probability of existence

The *probability of existence* (PEX) for an activity is the estimated probability at a point in the search that an activity will be present in a final solution to the problem. This probability is in the range $[0, 1]$ with 1 indicating that an activity will certainly be part of the solution and 0 indicating that it certainly will not. PEX is represented by the expected value of boolean demand variables such as were used by Baptiste and Le Pape [1]. While the boolean PEX variables themselves can only take on the value of 0 or 1, the expected value

of the PEX variable is on the interval [0, 1]. The expected PEX value is a representation of additional information about each activity which can then be incorporated into the heuristic commitment techniques based on texture measurements.

It is important to make clear the fact that the PEX variable is a standard boolean constrained variable: in any solution, each PEX variable must be assigned either to 0 or to 1. The expected PEX value, however, is a number in the interval [0, 1]. When a PEX variable is assigned, its expected value is equal to its assigned value. When unassigned, however, the expected PEX value varies according to the propagation rules presented below.

In order to simplify the implementation for our model of alternative activities, we have placed one major limitation on the PEX representation: if a XorNode has an expected PEX value of x, k upstream and l downstream links, the expected PEX value of each directly connected upstream node is x/k and the expected PEX value of each directly connected downstream node is x/l. The only exception to the rule of even division is when an upstream or downstream node has a PEX variable assigned to 0 or 1. If a neighboring, without loss of generality, upstream node has a PEX variable assigned to 0, it is removed from the graph: the expected PEX value at the XorNode is simply divided among the upstream nodes with unassigned PEX variables. Similarly, if the neighboring, without loss of generality, upstream node has its PEX variable assigned to 1, the PEX variables of all the other directly linked upstream nodes must be assigned to 0: the XorNode acts as if it only has a single upstream node. We discuss issues surrounding the removal of this limitation in Section 10.

4. Propagators for alternative activities

The power of propagators is well-recognized in the constraint-directed scheduling literature. It appears likely, therefore, that the use of propagators in problems with PEX variables will lead to improved search performance.

In this section, we discuss three types of propagation on the extended temporal network:

(1) PEX propagation—The expected PEX values of different nodes are related via the topology of the temporal network. When the network is modified (e.g., by a commitment that sets the PEX variable of an activity to 0), the effects of the commitment must be propagated to the expected PEX values of the other nodes in the network.

(2) Temporal propagation—Standard temporal propagation enforces arc-B-consistency [29] on the temporal constraints in the scheduling problem. For example, if A_i and A_j are directly connected via a precedence constraint such that A_j is a successor of A_i, temporal propagation enforces that $est_j \geqslant est_i + dur_i$ and $lft_i \leqslant lft_j - dur_j$. To account for nodes that do not necessarily exist in a final solution, we must adapt the standard temporal propagation algorithm.

(3) PEX-edge-finding propagation—Edge-finding propagators (both exclusion [12,13, 15,16,31] and not-first/not-last [2,3,12,13,15,31]) have been shown to significantly improve scheduling performance in a number of types of scheduling problems. We extend edge-finding here to incorporate activities that do not necessarily execute in a final solution.

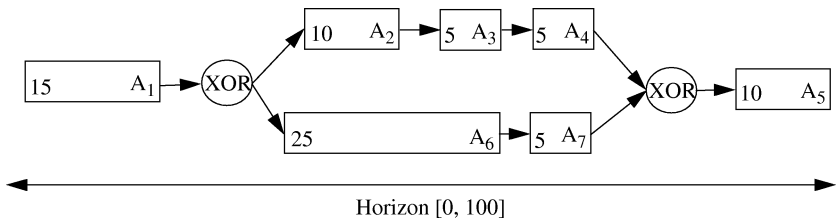


Fig. 5. A process plan with a choice of activities. The duration of each activity is shown in the lower left corner of the activity.

The small example in Fig. 5 serves to illustrate the first two types of functionality. The figure represents a single process plan with a choice of activities: either A_1 can be followed by A_2, A_3, A_4, and then A_5 or A_1 can be followed by A_6, A_7, and then A_5. The duration of each activity is indicated by the number in its lower left corner. Assuming that A_1 and A_5 must be executed, we assign their PEX variables to 1. Further, we split the probability of existence equally between the two choices. Therefore, A_2, A_3, A_4, A_6, and A_7 have expected PEX values of 0.5. The PEX propagation must ensure that if we were to make an arbitrary assignment of a PEX variable, such as assigning the PEX variable of A_3 to 1, the expected PEX values of the related activities and, perhaps, the PEX variables themselves, would be appropriately reset. In our example, the PEX variables of A_2 and A_4 should also be set to 1, and the PEX variables of A_6 and A_7 should be set to 0.

Returning to the original graph in Fig. 5 (i.e., before assigning the PEX variable of A_3 to 1), the standard temporal propagation algorithm would derive that the start time window of A_1 is [0, 45]. The latest start time of A_1 is found by the calculation of the longest path from A_1 to A_5 which happens to include A_6 and A_7. If A_1 were to start at time point 46, and A_6 and A_7 were to be executed, A_5 would end at time point 101, which is after the end of the horizon. The difference in this graph, however, is the presence of the alternative path from A_1 to A_5. It is possible for A_1 to start at time 46 if the path going through A_2, A_3, and A_4 is chosen. In fact, with this choice, A_1 can consistently start as late as time point 55. Clearly, then, we need to modify the temporal propagation (or the temporal network) to account for the possibility that some activities may not be present in the final solution. Furthermore, assume that the two alternatives in Fig. 5 are still possible and that through some other scheduling decision the earliest start time of A_1 is increased to 46. Temporal propagation should detect that such a change makes one of the alternative paths inconsistent: there is no way that A_6 and A_7 can execute.

4.1. Propagation of expected PEX values

The basic PEX propagation algorithm is achieved through the PEX propagation behavior at each temporal node. As described above, at an AndNode (or Activity), A, all the non-XorNodes directly linked to A must have the same expected PEX value as A. Therefore, when PEX propagation enters an AndNode, the local expected PEX value can be modified and the new local value is propagated to all neighboring nodes. For XorNodes, the sum of the expected PEX values for all nodes directly connected upstream must be equal to

the sum for all nodes directly connected downstream which in turn must be equal to the expected PEX value of the XorNode.

The basic PEX propagation algorithm presented here ignores the current expected PEX values and re-propagates the entire network. In practice, we also have an incremental algorithm that updates the expected PEX values after a local change. Details of the incremental algorithm can be found in Beck [4].

Given our temporal networks, if a node has either no upstream or no downstream links it cannot be subject to a XorNode and so must be present in the final solution. Any temporal node that either has no upstream links or no downstream links must, therefore, have its PEX variable assigned to 1. Also since a requirement of a legal temporal network is that the expected PEX values are unambiguous, initial propagation may begin with all nodes with no upstream links or all nodes with no downstream links. We have arbitrarily chosen to perform the initial propagation from all nodes with no upstream links. Recall that the expected PEX value of any node that has its PEX variable assigned is equal to the assigned value.

Pseudo-code for the initial propagation algorithm is given in Fig. 6. Lines 4, 8, and 17 make use of active nodes and active links. An *active link* is a binary temporal constraint such that the PEX variable of each of its nodes is not assigned to 0. An *active node* is a node connected via an active link. Recall that a topological sort (line 1) in an acyclic, directed network is an ordering of the nodes such that if there is a path from node Y to node X in the graph then $Y < X$ in the sort order [33]. In our case, the topological sort establishes that if Y is upstream of X in the temporal graph, then $Y < X$ in the sort order. In the loop beginning at line 2, therefore, we are guaranteed that all upstream nodes already have their expected PEX values assigned.

```
1:  create downstream topological sort of temporal nodes
2:  for each temporal node with unassigned PEX variable in sort
    order
3:    if the node is an AndNode OR Activity
4:      upstreamNode = any directly connected, active upstream node
5:      set ExpPEX = downstreamExpPEXValue (upstreamNode)
6:    if the node is an XorNode
7:      set ExpPEX = 0
8:      for upstreamNode = all directly connected, active upstream
        nodes
9:        set ExpPEX += downstreamExpPEXValue (upstreamNode)
10:
11:
12: procedure downstreamExpPEXValue (Node node)
13:
14:   if the node is an AndNode OR Activity
15:     return ExpPEX
16:   if the node is an XorNode
17:     return (ExpPEX / number active downstream links)
```

Fig. 6. Pseudocode for the basic PEX propagation algorithm.

The complexity of the initial propagation given n nodes and m temporal constraints is $O(\max(m, n))$. The topological sort (implemented with a depth-first search) and the for-loop are both $O(\max(m, n))$ as all nodes and constraints must be visited.

4.2. Temporal propagation with PEX

There are two differences in temporal propagation when XorNodes are present in the network.

(1) Propagation through a XorNode is different than propagation through an Activity.
(2) It is possible to derive a dead-end in the search by emptying the domain of a variable during temporal propagation. When there are nodes with unassigned PEX variables in the graph, such a state may not be a dead-end, but rather may indicate an implied PEX commitment.

4.2.1. Temporal propagation through a XorNode

The temporal propagation through a XorNode is straightforward given the temporal semantics of a XorNode. A XorNode enforces the temporal relationship that it must start at the same time as or after the end time of at least one of its upstream neighbors, and it must end at the same time as or before the start time of at least one of its downstream neighbors. During downstream propagation, therefore, the XorNode sets the lower-bound on its start time domain based on the minimum earliest finish time of its upstream neighbors. This start time is then propagated further downstream. Similarly, during upstream propagation the upper-bound on the end time of a XorNode is set based on the maximum latest start time of its downstream neighbors and this value is further propagated upstream.

4.2.2. Deriving implied PEX commitments from temporal propagation

In standard temporal propagation algorithms, if a variable's domain has been emptied, a dead-end is derived. When PEX variables are present, emptying a domain may not be a dead-end. Rather it may indicate that a particular PEX variable must have a value of 0.

When a domain is emptied in temporal propagation, the PEX variable of the temporal node with the emptied domain is examined. If the PEX variable is unassigned, the node is marked to indicate that the PEX variable has been determined to be 0 and temporal propagation does not continue from that node. After temporal propagation, a separate algorithm examines all the temporal nodes to determine if any have been marked to indicate the derivation of an implied PEX constraint. If such a temporal node is found, a new unary PEX constraint is asserted, and the usual PEX propagation and temporal propagation is done.

If the domain of a temporal node with PEX of 1 is emptied, a dead-end is derived as in the standard temporal propagation. To see why this is the case, imagine emptying the start time domain of an activity, A, with a PEX variable assigned to 1. By definition, it cannot have any enclosing XorNodes. Assume that the presence of some activity, B, with an unassigned PEX variable caused the empty domain. The temporal propagation from B to A must occur through one of the XorNodes enclosing B. If that propagation empties A, then it must be the case that not only B but *all* its alternatives are inconsistent with A. Therefore, it is a true dead-end. Note that this reasoning holds even if activity B is nested

inside a number of alternatives: the temporal propagation from B to A must pass through the outer-most XorNode enclosing B. This XorNode, X, must have a PEX variable equal to 1. Since all the alternatives of X are inconsistent with A, we have a true dead-end.

4.3. PEX-edge-finding propagation

Two types of edge-finding have been widely used in the constraint-directed scheduling literature [12,13,15,16,31]. The reasoning in edge-finding is based on examining the set of activities on a resource and deriving temporal constraints that enforce new upper and lower bounds on the end and start times of activities.

Clearly, edge-finding can be used with activities with PEX variables assigned to 1. Imagine the situation, however, where the set, S, of activities on a resource contains a single activity, A, whose PEX variable is not assigned. Further assume that edge-finding can infer no new commitments or dead-ends when only considering the activities in the set $S \setminus \{A\}$. When considering all activities in S:

(1) If edge-finding derives a dead-end, we can soundly infer that activity A cannot execute and therefore set the PEX variable of A to 0.
(2) If edge-finding derives new unary temporal constraints on A, they can be soundly asserted. Clearly if A is to execute, it must be consistent with the rest of the activities that execute on the same resource. Therefore, any unary temporal constraints on A are sound: they must be true if A is to execute and if A does not execute they do not affect any other activities.
(3) If edge-finding derives any unary temporal constraints on activities other than A, they must be discarded. Since it is possible for A not to execute, and since the

```
1: procedure PEXEdgeFinding
2:   list-of-commitments = EdgeFinding()
3:   if list-of-commitments is not empty
4:     return list-of-commitments
5:
6:   for each activity A such that 0 < A_PEX < 1
7:     temporarily set A_PEX = 1
8:     tmp-list = EdgeFinding()
9:     for each commitment in tmp-list
10:      if dead-end
11:        list-of-commitments.insert (set A_PEX to 0)
12:      else if new commitment on activity A
13:        list-of-commitments.insert (commitment)
14:      else
15:        discard commitment
16:    reset A_PEX
17:
18:  return list-of-commitments
```

Fig. 7. Pseudocode for PEX-Edge-finding.

temporal constraints are derived on the assumption that A would execute, we cannot soundly constrain the other activities.

This reasoning leads to the PEX-edge-finding algorithms presented in Fig. 7. Note that this function uses the usual edge-finding algorithms as sub-routines (line 2 and line 8). That is, the calls to `EdgeFinding()` are calls to the standard edge-finding functions, which return a list of commitments that are implied by the current search state. Given that the standard edge-finding worst-case complexity is $O(n^2)$, where n is the number of activities on a resource, the PEX-edge-finding worst-case complexity is $O(n^3)$ for one resource.

5. Incorporating alternative activities into scheduling heuristics

With the ability to represent and propagate the fact that some temporal nodes may not be present in a final schedule, it is necessary to extend the heuristic search techniques to allow direct reasoning about an activity's probability of existence. The obvious extension is to expand the type of commitments that a heuristic can make to include setting the PEX variable of an activity. This, indeed, is the basic extension to existing heuristics.

Before presenting our extensions to texture measurement-based heuristic commitment techniques, it is necessary to briefly present two texture-based heuristic commitment techniques from the literature.

5.1. Texture measurement-based heuristics

A *texture measurement* is an analysis of the constraint graph underlying a problem state in order to distill information that can be used by the heuristic commitment technique [22]. For example, *contention* [36,37] 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. Two texture-based heuristic commitment techniques, SumHeight [8,10] and VarHeight [4,7], are extended in this paper to address scheduling problems with alternative activities.

The SumHeight texture measurement estimates the contention on each resource, over time, by summing the probabilistic individual demand curves that each activity has for the resource. If an activity does not require a resource, its demand is 0. Otherwise, a uniform probability distribution over the possible start times of the activity is assumed: each start time has a probability of $1/|STD|$ where STD is the domain of the activity's start time variable. The individual demand, $ID(A, R, t)$, is the probabilistic amount of resource R, required by activity A, at time t. It is calculated as follows:

$$ID(A, R, t) = \sum_{t - dur_A < \tau \leqslant t} \sigma_A(\tau), \tag{1}$$

where

$$\sigma_A(\tau) = \begin{cases} \dfrac{1}{|STD_A|} & \text{if } \tau \in STD_A, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

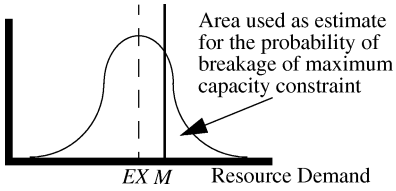


Fig. 8. Calculating the probability of breakage at event t with VarHeight.

The VarHeight texture measurement makes use of the same underlying individual demand curve as SumHeight. Rather than estimating aggregate demand, however, the VarHeight texture estimates the probability of breakage of a constraint based on the expected value of the aggregate demand and a distribution around the expected value. The expected value at each event point is provided by the aggregate curve on a resource as calculated by SumHeight. The distribution is created by the aggregation of the variance of the individual demands. In considering resource R, a time point t, and an activity A, we can associate a random variable X with the demand that A has for R at time t. For unary resources the domain of X is $\{0, 1\}$. The expected value for X, EX, assuming a uniform probability distribution for the start time of A, is $ID(A, R, t)$ as calculated in Eq. (1). We calculate the variance of X, VX, as follows (see [4] for the derivation):

$$VX = EX \times (1 - EX). \tag{3}$$

With a number of assumptions [4,7], two curves, representing the aggregate expected value and aggregate variance are formed by summation of the individual curves. Further assuming that the aggregate random variable is normally distributed around the aggregate expected value, the expected value and the standard deviation of the random variable define a distribution at time t. This is illustrated in Fig. 8. The area under the curve greater than the maximum capacity constraint is used as an estimate of the probability of breakage.

SumHeight and VarHeight texture measurements are both used to indicate the resource and time point with maximum criticality. In the case of SumHeight, criticality is defined using aggregate demand while for VarHeight, it is defined using the estimate probability of constraint breakage. A heuristic technique can then be used to derive a new commitment which attempts to decrease that maximum criticality. For example, in [10], the individual and aggregate curves created by the SumHeight texture are used to identify and sequence a pair of activities which contend for the critical resource at the critical time point.

5.2. Extending texture-based heuristics

The texture measurements presented above are based on an analysis of the constraint graph to identify the critical points where a commitment should be made, followed by a commitment that attempts to decrease that criticality. Expected PEX values represent potentially important search information and therefore should be embedded in the texture measurements to ensure that they are taken into account in the estimation of criticality at a search state.

The PEX variables affect texture-based heuristics in two ways. First, it is desirable to incorporate the expected PEX values into the underlying texture measurement estimation

technique. The criticality of an activity may be very different if it has a 0.125 probability of executing than if it has a 0.5 probability. Second, once textures have been calculated, the presence of PEX variables expands the types of heuristic commitments that can be made.

5.2.1. Adding PEX to texture curves

The sole modification to texture measurements to incorporate expected PEX values is to apply a vertical scaling factor to the individual activity demand. If the expected PEX value of activity A is 0.5, the individual demand at any time point t, $est_A \leqslant t < lft_A$, is half what the demand would be if the PEX variable were assigned to 1. This modification fits with our semantic interpretation of individual demand to be the probabilistic demand of an activity at a time point. Because we interpret an expected PEX value as an activity's probability of existence, an activity that has only a 50% likelihood of existing has half the probabilistic demand of an identical activity that will definitely exist. This is the same way that the probability of existence of an activity is taken into account in the KBLPS scheduler [14].

Recall that the individual demand, $ID(A, R, t)$, is (probabilistically) the amount of resource R, required by activity A, at time t as calculated in Eq. (1). The modification to this calculation that incorporates PEX, $ID_{PEX}(A, R, t)$ simply multiplies by A_{PEX}, the expected PEX value of activity A:

$$ID_{PEX}(A, R, t) = A_{PEX} \times ID(A, R, t). \tag{4}$$

Clearly, when the expected PEX value is equal to 1, ID_{PEX} is equal to ID.

5.2.2. New heuristic commitments

Independent of the texture measurement used, the basic texture algorithm remains the same (with ID_{PEX} replacing ID) up to the point where a commitment is chosen. That is, the individual demands are calculated and aggregated, and the resource and time point with highest criticality is identified. To incorporate PEX variables, we follow the same intuition as in our non-PEX, texture-based heuristics: make a commitment that will tend to most reduce the criticality of the most critical resource.

The procedure for generation of heuristic commitment techniques in the presence of PEX variables is as follows. We identify three activities:

(1) The activity, A, with unassigned PEX variable, A_{PEX}, and with the highest individual demand for the critical resource, R^*, at the critical time point, t^*, of all activities with unassigned PEX variables.
(2) The pair of activities, B and C, that are not currently sequenced, with PEX variables $B_{PEX} = 1$ and $C_{PEX} = 1$, and with the highest individual demand for the critical resource at the critical time point of all activities with a PEX variable assigned to 1. Without loss of generality, assume that the individual demand of activity B at the critical time point is greater than or equal to the individual demand of C at the critical time point.

The heuristic commitment is found by comparing the individual demand of A at t^* with that of B. If the individual demand of A is higher, then it is the most critical activity. Since A has a possibility of not existing, the heuristic decision to reduce criticality is to remove

A from the schedule by setting its PEX variable to 0. On the other hand, if B has the higher individual demand, then it is necessary to sequence B and C to attempt to reduce criticality.

If the heuristic PEX commitment is retracted via a complete retraction technique, we soundly post its opposite, setting the PEX variable of A to 1. Similarly, if the sequencing commitment is retracted, we post the opposite sequence.

The adaptation of the SumHeight and VarHeight heuristics by adding the PEX reasoning results in two new heuristic commitment techniques which we refer to as SH-PEX and VH-PEX, respectively.

5.2.3. Sequencing the critical activities

The final detail of the texture-based heuristics are the heuristics for sequencing activities B and C, that is, the pair of critical activities, both of which have a PEX variable assigned to 1. We use the sequencing heuristics presented in Beck [8,10].

5.3. Other heuristics

To form a basis of comparison for the quality of the texture-based heuristics for PEX, we modify other heuristics to incorporate the PEX variables. Here we present the modifications to PCP and LJ heuristics.

5.3.1. PCP-PEX

The *Precedence Constraint Posting* (PCP) heuristic [17,42] identifies the pair of activities on the same resource that are not sequenced and that have the minimum biased-slack measurement. These two activities are sequenced to preserve the maximum amount of slack. To adapt the PCP heuristic to activities with PEX variables, we follow the intuition that a commitment should preserve the maximum amount of slack.

The adaptation first specifies that the biased-slack measurement is calculated only for activity pairs such that neither activity has its PEX variable assigned to 0. The following three conditions then apply to the activity pair with the smallest biased-slack:

(1) If both activities have a PEX variable assigned to 1, follow the PCP heuristic and post the sequencing constraint that preserves the most slack.
(2) If one activity, A, has a PEX variable assigned to 1 and the other, B, has an unassigned PEX variable, the greatest amount of slack will be preserved by setting the PEX variable of B to 0. B will be completely removed from competition with A thereby maximizing the resulting slack.
(3) If both activities have unassigned PEX variables, the greatest amount of slack will be preserved by setting the PEX variable of the activity with the longest duration to 0. By removing the activity with maximum duration, we maximize the resulting slack between the critical activity pair.

If any of these commitments is retracted, we can post its opposite (the other sequence for case (1), or setting the PEX variable to 1 in cases (2) and (3)) to guarantee a complete search.

5.3.2. LJ-PEX

The *Left-Justified Random* (LJ) heuristic [31,32], finds the smallest earliest finish time of all the unscheduled activities and then identifies the set of activities which are able to

start before this time. One of the activities in this set is selected randomly and scheduled at its earliest start time. When backtracking, the alternative commitment is to update the earliest start time of the activity to the minimum earliest finish time of all other activities on that resource.

Our modification of LJ to incorporate PEX variables, LJ-PEX, performs the following steps:

(1) Find the smallest earliest finish time of all activities with PEX variable not assigned to 0.
(2) Identify the set of activities with PEX variable not assigned to 0 that can start before the minimum earliest finish time.
(3) Randomly select an activity, A, from this set.
(4) Assign A to start at its earliest start time and assign its PEX variable, A_{PEX}, to 1.

The alternative commitment, should backtracking undo the commitment on activity A, is to update the earliest start time of A to the minimum earliest finish time of all other activities with the PEX variable not assigned to 0, on the same resource as A. Note that the alternative does not contain a PEX commitment, which means that subsequent heuristic and implied commitments can still assign A_{PEX} to either 1 or 0. This ensures completeness of the search. With chronological backtracking, if the commitment on activity A is undone, it has been derived that A cannot start at its earliest start time in any schedule. The alternative then is that A starts later or that A does not execute at all. For a complete search we need to preserve these two alternatives when backtracking.

5.4. The information content of heuristic commitment techniques

One of the key differences among the heuristics that incorporate PEX is the extent of that incorporation. The texture-based heuristics use the expected PEX value to calculate the underlying individual demand of an activity and so it has a deep impact on the heuristic commitments that are made. LJ-PEX and PCP-PEX, in contrast, only use the PEX variable, not its expected value: these heuristics do not use the fact that one activity may have an expected PEX value of 0.125 while another an expected PEX value of 0.5. We predict that, because the texture-based heuristics take into account the information represented by the expected value, they will outperform heuristics that do not use that information.

6. Empirical evaluation

The empirical evaluation of the PEX techniques in this paper focuses on problems with alternative process plans and problems with both alternative process plans and alternative resources. The PEX techniques presented are applicable, without extension, to both types of problems as an activity with alternative resources is simply treated as a nested alternative within a process plan.

6.1. Experimental design

The primary purpose of the experimental evaluation is to determine the efficacy of using expected PEX values as part of the information underlying heuristic commitments. As

noted above, among the heuristic commitment techniques, only SH-PEX and VH-PEX use the expected value of the PEX variable in each search state to inform their decision making. An important component of the empirical evaluation is to determine if using this extra information results in better heuristic commitments and overall search performance. A second purpose is the evaluation of the PEX-edge-finding techniques. Given the relatively high time-complexity of the propagators, we want to evaluate their efficacy in terms of overall search performance.

Two of the obvious interesting parameters for alternative process plan problems is the number of alternatives in each process plan and the size of the overall scheduling problems. In Experiment 1 we look at the former while examining the latter in Experiment 2. Both experiments use a fully crossed design with makespan factor as the second independent variable.

The overall pattern we predict in Experiment 1 is that as the number of alternatives increases, the requirement for direct reasoning about the alternatives will become more important to good algorithmic performance. For Experiment 2, we focus on a more global factor of difficulty: the problem size. Our primary interest is to see how the algorithms with higher complexity components (such as PEX-edge-finding) are able to scale in terms of problem solving ability in relation to the lower complexity algorithms.

In our final experiment, Experiment 3, we incorporate alternative resources into the alternative process plan problem sets from Experiment 1 (see Section 8.1.1 for a detailed description of this incorporation). The primary impact of the addition of alternative resources is to greatly increase the range of expected PEX values across the activities in the problem. As with the increase in the number of alternatives per process plan, we predict the increase in the range of expected PEX values to favor the heuristics that make detailed use of them in their heuristic commitment techniques.

6.2. The reporting of time-outs

The experiments in this paper 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. A time-out indicates that the algorithm was unable to find a solution or prove that no solution exists for a particular scheduling problem instance.

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 [24,25] 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 [9], 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 varies, 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.

The use of time-outs as a search statistic allows us to integrate search performance on over-constrained problems and soluble problems into a single statistic. The intuition is that algorithms fail when they can neither find a solution nor a proof of insolubility. By using the number of failures, in this way, we get a clearer picture of the algorithm performance. For example, plotting the number of problems for which a solution is found obscures the fact that some algorithms may be performing very well on over-constrained problems (by finding proofs of insolubility) whereas others are not able to find any such proofs.

6.3. Summary of algorithms

Eight algorithms are used in the experiments. Each is an instantiation of the ODO constraint-directed scheduling framework [5].

All algorithms use chronological backtracking and therefore can each be specified by identifying the heuristic commitment and set of propagators used. Two sets of propagators are used:

(1) PEX Propagators: Temporal propagation, PEX-edge-finding exclusion, PEX-edge-finding not-first/not-last, and Constraint-Based Analysis (CBA).
(2) Non-PEX Propagators: Temporal propagation, Edge-finding exclusion, Edge-finding not-first/not-last, and Constraint-Based Analysis (CBA).

Table 3 presents the names by which we refer to each algorithm in the balance of this paper.

In our experiments, each algorithm is run until it finds a solution or until a 20 minute CPU time limit has been reached in which case failure is reported. The machine for all experiments is a Sun Ultra-Sparc-IIi, 270 MHz, 128 M memory, running SunOS 5.6.

6.4. Performance measures

A number of performance measures are reported:

- Fraction of problems timed-out—A primary measure of performance is the fraction of problems that each algorithm was neither able to find a solution nor show that none exists within the experimental time-limit.
- Mean CPU time—For each problem set, we report the mean CPU time for each algorithm.

Table 3
The eight algorithms used in the alternative process plan experiments (see the text for the components of the propagator sets)

Algorithms		
Heuristic	With PEX propagators	With non-PEX propagators
SH-PEX	SH-PEX-P	SH-PEX-NP
VH-PEX	VH-PEX-P	VH-PEX-NP
PCP-PEX	PCP-PEX-P	PCP-PEX-NP
LJ-PEX	LJ-PEX-P	LJ-PEX-NP

- Mean number of backtracks—We also report the mean number of times that an algorithm backtracked while searching for a solution to the problems at a point in the parameter space.
- Mean number of heuristic commitments—The mean number of commitments found by the heuristic commitment technique. A heuristic commitment is a commitment made by a heuristic technique. Unlike implied commitments, heuristic commitments can be mistaken and therefore represent a branching in the search tree.
- Mean number of implied commitments—The mean number of commitments found by the propagators. An implied commitment is a constraint that is added to the constraint graph because it is already implied by the problem state. Implied commitments include those found by all propagators with the exception of temporal propagation. The temporal propagation commitments are omitted because they are implemented as the removal of possible values from the domains of the start- and end-time variables rather than the explicit addition of a constraint.
- Mean number of total commitments—The mean total number of commitments (found by the heuristic commitment technique and the propagators).

When an algorithm times-out on a problem instance, the performance measures for that problem are included in the calculation of the mean. While this inclusion means that our results represent a lower-bound on performance, it avoids skewing results based on discarding data for a problem that some algorithm times-outs on.

Unless otherwise noted, statistical significance for each performance measure is evaluated with the boot-strap paired-t test [18] with $p \leqslant 0.0001$.

Our interest in these experiments is the comparison of the four heuristic commitment techniques (SH-PEX, VH-PEX, PCP-PEX, and LJ-PEX) and an evaluation of the efficacy of PEX-edge-finding. The statistical analysis therefore compares the four heuristics with each other in two conditions: with and without PEX-edge-finding. To address the question of the usefulness of PEX-edge-finding, we also compare its use in four conditions corresponding to each of the heuristics. A summary of these groups is shown in Table 4.

7. Alternative process plans

7.1. Experiment 1: Scaling with the number of alternatives

7.1.1. Problems

We expect that the number of alternatives in each process plan will have an impact on the search performance. To examine algorithm performance under varying numbers of alternatives, we constructed four problem sets with varying maximum numbers of alternatives in each process plan. To illustrate this construction we use a problem set with a maximum of three alternatives.

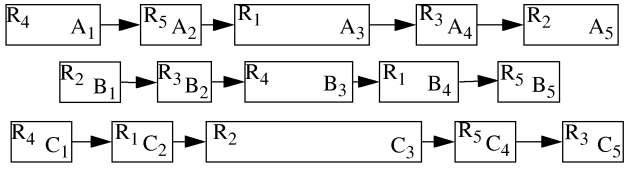
Each problem begins with an underlying job shop problem. All problems in this experiment have 10 activities per process plan and 10 resources. For a problem with a maximum of three alternatives per process plan, we generate 30 jobs of 10 activities each. These 30 jobs are then transformed into 10 process plans with alternatives. For each job we randomly choose the number of alternatives, k, it will have uniformly from the

Table 4
The groups of algorithms used in the statistical tests

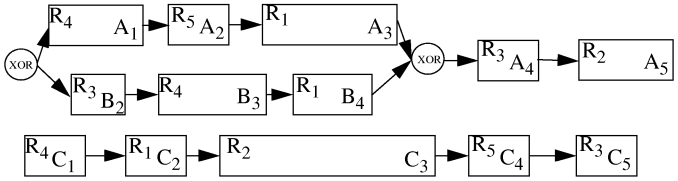
Comparison group	Purpose
SH-PEX-P	
VH-PEX-P	Compare heuristics when used with PEX-edge-finding
PCP-PEX-P	
LJ-PEX-P	
SH-PEX-NP	
VH-PEX-NP	Compare heuristics when used without PEX-edge-finding
PCP-PEX-NP	
LJ-PEX-NP	
SH-PEX-P	Evaluate PEX-edge-finding with SH-PEX heuristic
SH-PEX-NP	
VH-PEX-P	Evaluate PEX-edge-finding with VH-PEX heuristic
VH-PEX-NP	
PCP-PEX-P	Evaluate PEX-edge-finding with PCP-PEX heuristic
PCP-PEX-NP	
LJ-PEX-P	Evaluate PEX-edge-finding with LJ-PEX heuristic
LJ-PEX-NP	

interval $[0, 3]$. We then combine randomly chosen portions of the next k jobs with our original job to produce a single process plan with k alternatives. The combination process randomly chooses the to-be-combined portion and the location in the original process plan where the alternative is to be inserted. The only requirements are that each path between a pair of XorNodes representing an alternative must have the same number of activities and that number must be greater than 1. The latter requirement avoids problem structures that are better categorized as alternative resources while the former requirement avoids situations where an entire ten-activity process plan can have a two-activity alternative. Such a situation would lead to scheduling problems with essentially a single easy decision to make (choose the shorter process plan). We prefer to generate harder problems and so avoid such situations. Fig. 9 illustrates the combination of three process plans into a single process plan with two alternatives.

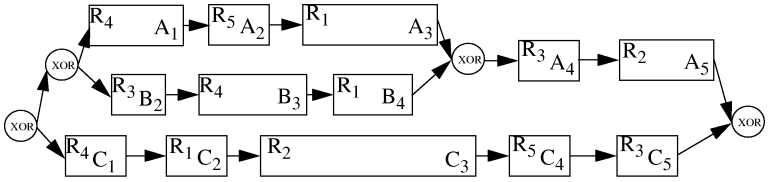
Sets of problems with a maximum of one, three, five, and seven alternatives per process plan were generated. Each set contains 20 problems. For each problem the job lower bound was calculated taking into account the alternatives. This calculation finds the shortest path (where path length is determined by the sum of the durations of activities on a path) in each job. We then use a makespan factor spanning 1.0 to 1.5 (in steps of 0.1) to generate a total of six problem sets of 20 problems each for each number of alternatives.



Original Process Plans



Process Plans After First Combination (B_1 and B_5 are discarded)



Final Process Plans (After Second Combination)

Fig. 9. Generating a single process plan with two alternatives.

Table 5
The characteristics of the problems in Experiment 1

Problem set	Number of activities per problem		
(Max. # of alternatives)	Minimum	Mean	Maximum
1	116	131.6	156
3	142	171.8	200
5	172	204.7	251
7	167	224.2	280

One of the results of this method of problem generation is that a solution to a problem will have exactly the same number of activities as the underlying job shop problem. Prior to scheduling, however, each problem has a different number of activities depending on the randomly chosen number and size of alternatives. Table 5 shows the characteristics of the problem sets in terms of the number of activities in the problem definition.

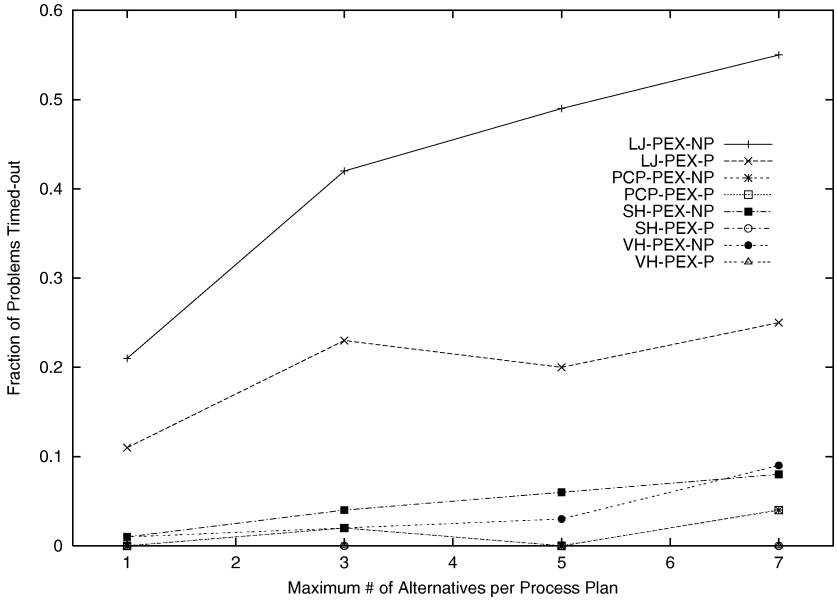


Fig. 10. The fraction of problems in each problem set for which each algorithm timed-out.

7.1.2. Results

The proportion of the problems in each set for which each algorithm times-out is shown in Fig. 10. Slightly obscured by the graph is the result that SH-PEX-P and VH-PEX-P do not time-out on any problems across all problem sets, and that there is no difference between PCP-PEX-P and PCP-PEX-NP. Statistical analysis indicates that regardless of the use of PEX-edge-finding, the algorithms using VH-PEX, SH-PEX, and PCP-PEX each time-out on significantly fewer problems than the corresponding algorithm using LJ-PEX. With PEX-edge-finding there are no significant differences among SH-PEX-P, VH-PEX-P, and PCP-PEX-P, while without PEX-edge-finding the only significant difference not involving LJ-PEX is that PCP-PEX-NP times-out on significantly fewer problems than SH-PEX-NP ($p \leqslant 0.0005$). In terms of the usefulness of PEX-edge-finding, SH-PEX-P, VH-PEX-P, and LJ-PEX-P time-out on significantly fewer problems than SH-PEX-NP, VH-PEX-NP, and LJ-PEX-NP respectively. There is no significant difference in performance between PCP-PEX-NP and PCP-PEX-P.

Turning to mean CPU time, Fig. 11 displays the mean CPU time across all problem sets. Overall, there is no significant difference between SH-PEX-P and VH-PEX-P. Both algorithms use significantly less mean CPU time than PCP-PEX-P, which in turn uses significantly less mean CPU time than LJ-PEX-P. When PEX-edge-finding is not used, there is no difference among SH-PEX-NP, VH-PEX-NP, and PCP-PEX-NP, but all three are significantly better than LJ-PEX-NP.

Holding the heuristic component constant, we see that SH-PEX-P, VH-PEX-P, and LJ-PEX-P all incur a lower mean CPU time than their corresponding non-PEX-edge-finding algorithms, while there is no difference between PCP-PEX-P and PCP-PEX-NP.

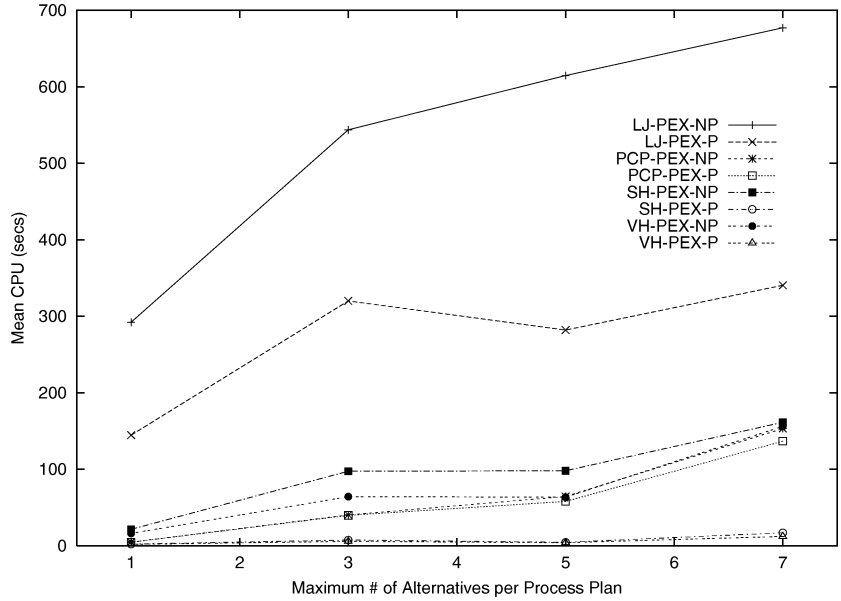


Fig. 11. The mean CPU time in seconds for each problem set.

Other search statistics

Other search statistics reveal the following results:

- VH-PEX-P makes significantly fewer backtracks ($p \leqslant 0.001$), commitments ($p \leqslant 0.005$), and heuristic commitments ($p \leqslant 0.0005$) than SH-PEX-P. SH-PEX-P makes significantly fewer backtracks, commitments, and heuristic commitments than PCP-PEX-P which in turn makes significantly fewer backtracks, commitments, and heuristic commitments than LJ-PEX-P.
- VH-PEX-NP, SH-PEX-NP, and PCP-PEX-NP each make significantly fewer backtracks, commitments, and heuristic commitments than LJ-PEX-NP. In addition, VH-PEX-NP makes significantly fewer backtracks ($p \leqslant 0.005$) and heuristic commitments ($p \leqslant 0.005$) than SH-PEX-NP. The only other significant differences are in the overall commitments, where VH-PEX-NP makes significantly fewer than ($p \leqslant 0.005$) SH-PEX-NP which in turn makes significantly fewer than PCP-PEX-NP ($p \leqslant 0.005$).
- With each heuristic, the use of PEX-edge-finding results in significantly fewer backtracks and heuristic commitments ($p \leqslant 0.0005$ when PCP-PEX is the heuristic). In terms of total commitments, LJ-PEX-P is not significantly different from LJ-PEX-NP, while the difference is significant for the other three heuristic commitment techniques ($p \leqslant 0.005$ for PCP-PEX-P versus PCP-PEX-NP).

7.1.3. Summary

Experiment 1 indicates that:

- The LJ-PEX heuristic when used with or without PEX-edge-finding performs significantly worse than the other heuristics across all the problem sets.

Table 6
The characteristics of the problems in Experiment 2

Problem set	Number of activities per problem		
(Size of underlying jobshop problem)	Minimum	Mean	Maximum
5×5	36	61.7	85
10×10	172	204.7	251
15×15	356	430.9	524
20×20	652	738.8	857

- While there is no significant difference in terms of the number of problems timed-out among SH-PEX-P, VH-PEX-P, and PCP-PEX-P, all other search statistics indicate VH-PEX-P and SH-PEX-P perform significantly better than PCP-PEX-P.
- There is little performance difference among VH-PEX-NP, SH-PEX-NP, and PCP-PEX-NP.
- PEX-edge-finding improves scheduling performance when used with the VH-PEX, SH-PEX, and LJ-PEX heuristics. No such improvement was found with PCP-PEX heuristic.

7.2. Experiment 2: Scaling with problem size

The purpose of Experiment 2 is to examine the scaling behavior of the algorithms as the problem size gets larger, but the number of alternatives remains fixed.

7.2.1. Problems

The problems used in this experiment are generated in the same way as the problems in Experiment 1. The difference is that rather than changing the maximum number of alternatives in different problem sets, we hold that parameter fixed at five while varying the size of the underlying job shop problem. All problems in Experiment 1 had a 10×10 underlying job shop problem. For this experiment, we look at problems whose underlying job shop problems are of sizes 5×5, 10×10, 15×15, and 20×20. The problem set from Experiment 1 with five alternatives is used again in Experiment 2. As with Experiment 1, we use the job lower bound and varying makespan factors to create six sets of 20 problems each at each problem size.

Due to the problem generation, different problem instances of the same size problem have slightly varying numbers of activities. Table 6 shows the characteristics of the problem sets.

7.2.2. Results

The proportion of problems timed-out for each algorithm is shown in Fig. 12. SH-PEX-P and VH-PEX-P time-out on significantly fewer problems than PCP-PEX-P which in turn times-out on significantly fewer problems than LJ-PEX-P. Without PEX-edge-finding, the results are slightly different as PCP-PEX-NP times-out on significantly ($p \leqslant 0.001$) fewer problems than SH-PEX-NP which in turn times-out on significantly

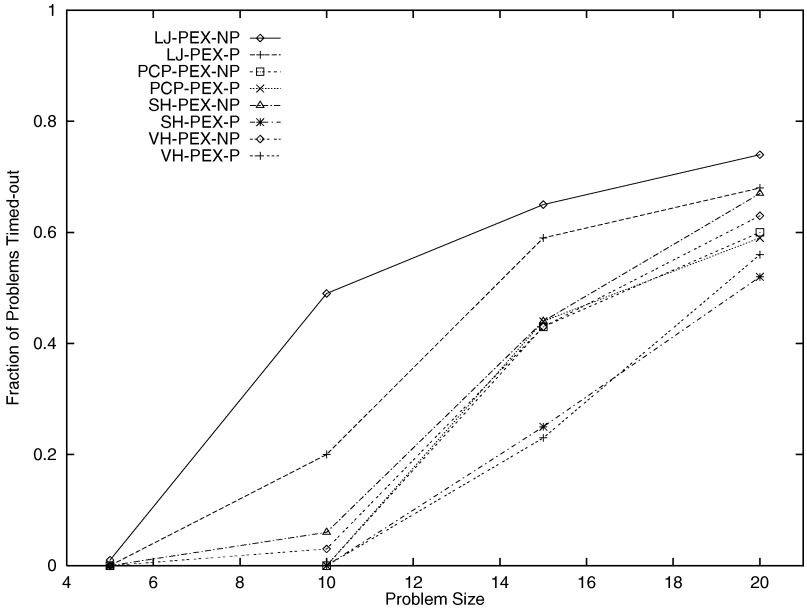


Fig. 12. The fraction of problems in each problem set for which each algorithm timed-out.

fewer problems than LJ-PEX-NP. VH-PEX-NP times-out on significantly fewer problems than LJ-PEX-NP, but there are no significant differences between VH-PEX-NP and SH-PEX-NP, or between VH-PEX-NP and PCP-PEX-NP. PCP-PEX-P shows no significant difference when compared to PCP-PEX-NP while SH-PEX-P, VH-PEX-P, and LJ-PEX-P time-out on significantly fewer problems than their respective non-PEX-edge-finding algorithms.

Turning to the mean CPU time results, the overall results are shown in Fig. 13. SH-PEX-P and VH-PEX-P each incur significantly less CPU time than PCP-PEX-P which in turn incurs significantly less CPU time than LJ-PEX-P. Without PEX-edge-finding, however, PCP-PEX-NP uses significantly less CPU time than SH-PEX-NP ($p \leqslant 0.0005$) which in turn uses significantly less CPU time than LJ-PEX-NP. There are no significant overall differences between VH-PEX-NP and SH-PEX-NP, or between VH-PEX-NP and PCP-PEX-NP. Evaluation of PEX-edge-finding shows, overall, that using it results in significantly lower CPU time with SH-PEX, VH-PEX, and LJ-PEXP, but not with PCP-PEX.

Other search statistics

A summary of the other search statistics is as follows:

- SH-PEX-P and VH-PEX-P each make significantly fewer backtracks, commitments, and heuristic commitments than both PCP-PEX-P and LJ-PEX-P. There are no significant differences in these statistics between PCP-PEX-P and LJ-PEX-P, or between SH-PEX-P and VH-PEX-P.

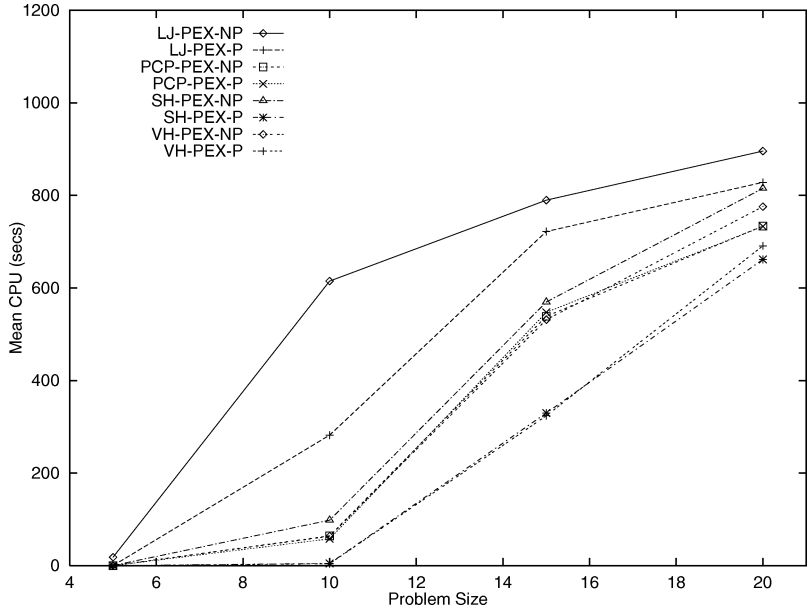


Fig. 13. The mean CPU time in seconds for each problem set.

- VH-PEX-NP makes significantly fewer backtracks ($p \leqslant 0.001$) and heuristic commitments ($p \leqslant 0.005$) than SH-PEX-NP while making significantly fewer backtracks, commitments, and heuristic commitments than either PCP-PEX-NP or LJ-PEX-NP. SH-PEX-NP and PCP-PEX-NP each make significantly fewer backtracks and heuristic commitments than LJ-PEX-NP. In addition, SH-PEX-NP makes fewer commitments than either PCP-PEX-NP or LJ-PEX-NP. There are no other significant differences among the non-PEX-edge-finding algorithms.
- The algorithms using PEX-edge-finding all make significantly fewer backtracks and heuristic commitments than their counterparts that do not use the propagator. In terms of total commitments, however, the only significant difference is that PCP-PEX-P makes significantly fewer than PCP-PEX-NP.

7.2.3. Summary

The results of Experiment 2 indicate that:

- With PEX-edge-finding, the algorithms using the VH-PEX and SH-PEX heuristics outperform the one using PCP-PEX which in turn outperforms the one using LJ-PEX.
- Without PEX-edge-finding, however, the relative performance changes: PCP-PEX outperforms SH-PEX which in turn outperforms LJ-PEX. VH-PEX shows few significant differences with either SH-PEX or PCP-PEX while significantly outperforming LJ-PEX.
- PEX-edge-finding typically results in better overall performance when used with SH-PEX, VH-PEX, and LJ-PEX. There is little difference between PCP-PEX-P and PCP-PEX-NP.

8. Combining alternative process plans and alternative resources

In our final experiment, we look at problems containing both alternative process plans and alternative resources.

8.1. Experiment 3: Scaling with the number of alternatives

8.1.1. Problems

All problems for this experiment are transformations of the problems used in Experiment 1. In those problems, each activity has only one resource alternative and a single possible duration.

Alternative resources were added to each activity by randomly generating the total number of resource alternatives following the distribution shown in Table 7. The original resource requirement and duration on that resource are preserved in the new problem. In addition, the new resource alternatives (if any) are randomly chosen with uniform probability from among all the other resources in the problem. The duration of the activity on each new alternative resource is generated by multiplying the activity's original duration by a randomly chosen factor in the domain $[1.0, 1.5]$ and then rounding to the nearest integer value. Note that an activity that is specified to have one alternative resource has only one resource on which it can execute: the original resource from the job shop problem.

These transformations result in problems such that:

- The original job lower bound calculated in Experiment 1 is still a valid lower bound. All alternative resources require the activity to have at least as large a duration as in the original problem; therefore, the shortest path in a job, including resource alternatives, remains the same.
- There is likely to be widely varying expected PEX values between activities. The theoretical range on the expected PEX value of an activity, A, at the beginning of a problem with seven alternatives is shown in Expression (5). The minimum expected PEX value is possible for an activity that represents one of six possible resource alternatives while the original activity (without resource alternatives) was nested inside the seven process plan alternatives. Widely ranging expected PEX values represent non-uniformities in problem structure: an activity with a high expected PEX

Table 7
The distribution of alternative resources for the problems in Experiment 3

Number of alternative resources per activity	Probability
1	0.03125
2	0.5
3	0.25
4	0.125
5	0.0625
6	0.03125

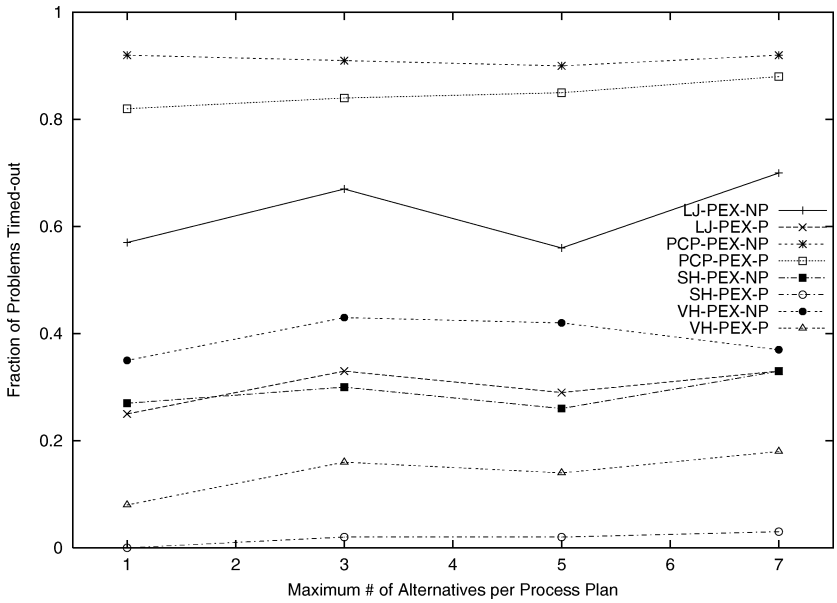


Fig. 14. The fraction of problems in each problem set for which each algorithm timed-out.

value is far more likely to execute than an activity with a low expected PEX value. We would predict, therefore, that heuristics that reason explicitly about the expected PEX value (i.e., SH-PEX and VH-PEX as opposed to the other heuristic commitment techniques) will be able to make higher quality commitments.

$$\tfrac{1}{3} \times 2^{-8} \leqslant A_{PEX} \leqslant 1. \tag{5}$$

8.1.2. Results

The fraction of problems in each problem set that each algorithm timed-out on is displayed in Fig. 14. These results indicate, regardless of the use of PEX-edge-finding, that SH-PEX outperforms VH-PEX which is better than LJ-PEX which in turn outperforms PCP-PEX. In addition, each heuristic times-out on significantly fewer problems when using PEX-edge-finding than without it.

The results for mean CPU time are displayed in Fig. 15. These results are consistent with the timed-out results on all accounts. Comparing heuristics shows that SH-PEX has a significantly lower mean CPU time than VH-PEX. VH-PEX incurs significantly less CPU time than LJ-PEX which in turn has a significantly lower mean CPU time than PCP-PEX. The PEX-edge-finding results indicate that each heuristic incurs a significantly lower mean CPU time when PEX-edge-finding is used.

Other search statistics

The other search statistics evaluated (number of backtracks, number of commitments, and number of heuristic commitments) agree in the relative ranking of the performance of each heuristic: regardless of the PEX-edge-finding condition, SH-PEX significantly

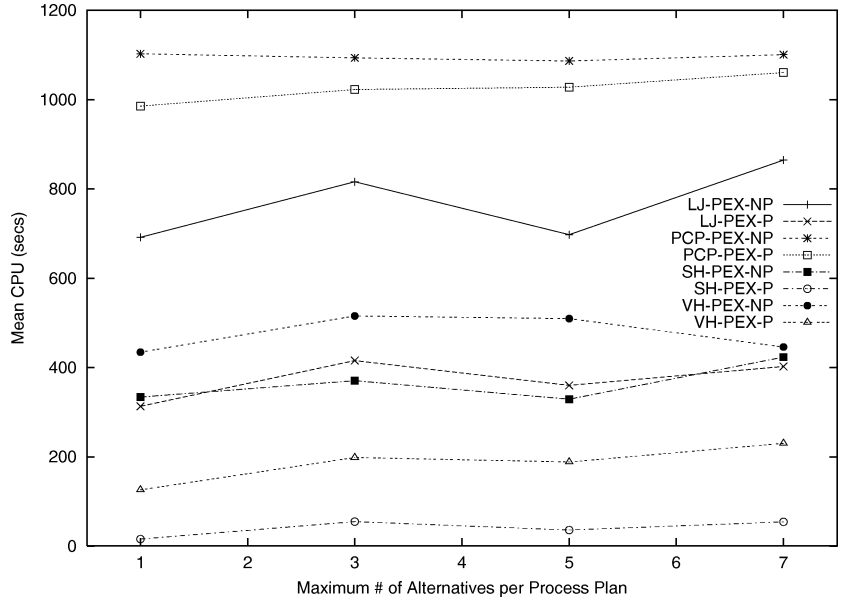


Fig. 15. The mean CPU time in seconds for each problem set.

outperforms VH-PEX which significantly outperforms LJ-PEX which in turn significantly outperforms PCP-PEX. The only exception to this pattern is in comparing the number of heuristic commitments made by LJ-PEX and PCP-PEX where there is no significant difference.

All heuristics exhibited significantly fewer backtracks and fewer heuristic commitments when used with PEX-edge-finding. Interestingly, VH-PEX, PCP-PEX, and LJ-PEX all made significantly more overall commitments when using PEX-edge-finding than when not using it. There is no significant difference in overall commitments between PEX-edge-finding conditions when SH-PEX is the heuristic commitment technique.

8.1.3. Summary

Experiment 3 strongly indicates that, independent of the propagator condition, SH-PEX outperforms VH-PEX which outperforms LJ-PEX which in turn outperforms PCP-PEX. In addition, PEX-edge-finding results in better performance regardless of the heuristic used.

9. Discussion

The problem structure hypothesis that we set out to investigate in the paper states that as a scheduling problem becomes more complex, a deeper understanding of the problem structure is necessary for successful heuristic search [41]. Overall our results support this hypothesis as it was the heuristics that exploited the PEX information that achieved superior performance. Support for the problem structure hypothesis is especially clear when the results of Experiment 3 are contrasted with those of Experiments 1 and 2. While

the PEX-based heuristics (SH-PEX and VH-PEX) achieved performance that equaled or bettered all other heuristics in the first two experiments, in the final experiment, their performance was dramatically superior. The central manipulation to produce the different problem structure in Experiment 3 was the more widely ranging expected PEX values. Because this added complexity is represented in the expected PEX values and because SH-PEX and VH-PEX take the expected PEX values into account in their problem structure analysis, the PEX-based heuristics achieve significantly better performance: as the problem complexity increases, knowledge of the problem structure is increasingly important for successful heuristic search.

9.1. Heuristics

Overall, while SH-PEX and VH-PEX outperform PCP-PEX with PEX-edge-finding, especially in Experiment 3, their comparison without PEX-edge-finding is less clear-cut. In Experiment 1 there are few differences; in Experiment 2, PCP-PEX is superior to SH-PEX; in Experiment 3 SH-PEX is superior to VH-PEX which is superior to PCP-PEX. LJ-PEX is outperformed by all other heuristics (regardless of the use of PEX-edge-finding) in Experiments 1 and 2, but outperforms PCP-PEX (again regardless of the use of PEX-edge-finding) in Experiment 3.

Experiment 3 clearly demonstrates the result of exploiting the extra information represented by the expected PEX values in the texture measurement upon which the heuristic commitment technique is based. The widely ranging expected PEX values represent a non-uniformity in the problem structure that is successfully exploited by the SH-PEX and VH-PEX heuristics. In the other experiments, though the range of expected PEX values is not as wide, SH-PEX-P and VH-PEX-P are still the best scheduling algorithms of the ones tested.

The comparison between PCP-PEX and SH-PEX is interesting:

- Experiment 1 shows no difference regardless of the use of PEX-edge-finding.
- Experiment 2 shows that PCP-PEX is better than SH-PEX without PEX-edge-finding, but worse with PEX-edge-finding.
- Experiment 3 shows that PCP-PEX is much worse than SH-PEX both with and without PEX-edge-finding.

These results demand an explanation both for why PCP-PEX performs so poorly in Experiment 3 and why it performs so well in Experiments 1 and 2.

We know, based on the results of [10], that on some job shop scheduling problem sets without resource-level non-uniformities, PCP is able to outperform SumHeight. No such non-uniformities were examined in this paper as our primary interest was the evaluation of reasoning about alternatives. A possible explanation for the good performance of PCP-PEX is that no resource-level non-uniformity exists for our experimental problem sets; therefore, the heuristic that does not specifically look for such non-uniformity achieves better overall performance.

A second explanation for the good quality of PCP-PEX is the quality of the underlying biased-slack heuristic. When there are no resource-level non-uniformities, a heuristic based on the identification of minimum-slack activity pairs and the subsequent commitment to maintain as much slack as possible has been shown to perform well in job shop scheduling.

Some of the quality of that heuristic is maintained when PEX is added. Again, in the absence of resource-level non-uniformities, a heuristic that identifies the pair of activities with the smallest time in which they may be sequenced and seeks to maximize that time (through a PEX commitment or a sequencing commitment) seems to be a reasonable heuristic to establish both a selection of activity alternatives and a sequencing of activities that satisfies the resource constraints.

Still the results of Experiment 3, especially the relative performance of PCP-PEX and LJ-PEX, demand explanation. We believe that these results are due to PCP's degraded ability to identify critical activity pairs in the presence of PEX values. In the original PCP heuristic, only activity pairs for which Constraint-Based Analysis (CBA) propagation has not inferred a sequence are examined to find the pair with minimum biased-slack [17,42]. When at least one member of an activity pair has an unassigned PEX variable, however, the CBA propagator cannot infer a sequence and therefore, regardless of the activity time windows, the biased-slack of the activity pair is evaluated. Activity pairs with little overlap will tend to have a very low biased-slack and therefore the heuristic will tend to focus on such activity pairs. To take the extreme case of an activity pair with disjoint time windows, the activities are not actually competing with each other for a resource reservation and so can hardly be a truly critical pair, yet they have a very small biased-slack. Furthermore, the commitment to set the PEX value of one of the activities to 0 is made on the intuition that, if such a commitment is not made, there is a high likelihood that the two activities will later over-capacitate the resource. But this cannot be the case with disjoint activities, and the likelihood that an activity pair with a small overlap will lead to an over-capacity is relatively small. This reasoning points to a modification of the PCP-PEX heuristic to calculate the biased-slack only for those activity pairs that would not have an implied sequence if they both had a PEX variable assigned to 1. Such a modification remains as future work.

9.2. PEX-edge-finding

Through almost all experiments and experimental conditions, PEX-edge-finding was shown to be beneficial to the overall problem solving ability of an algorithm. The only exception is when the PCP-PEX heuristic is used: little difference was seen between the PCP-PEX-P and PCP-PEX-NP except in Experiment 3 where PEX-edge-finding proved beneficial.

In general, these results are as expected. Given the significant increase in the performance of scheduling algorithms with the use of edge-finding propagators [31], we expect that some gain is likely with the PEX-edge-finding variation. Our intuitions as to why PEX-edge-finding (and indeed propagation, in general) should improve search performance rests on two impacts of propagation. First, propagation techniques reduce the search space by removing alternatives that would otherwise have to be searched through. Second, propagators improve the information upon which heuristics can be based. PEX-edge-finding improves both the information represented in the expected PEX values (by pruning inconsistent activity alternatives) and the information represented in the time windows of activities (because it infers unary temporal constraints on activities with unassigned PEX variables). SH-PEX and VH-PEX benefit from both improvements.

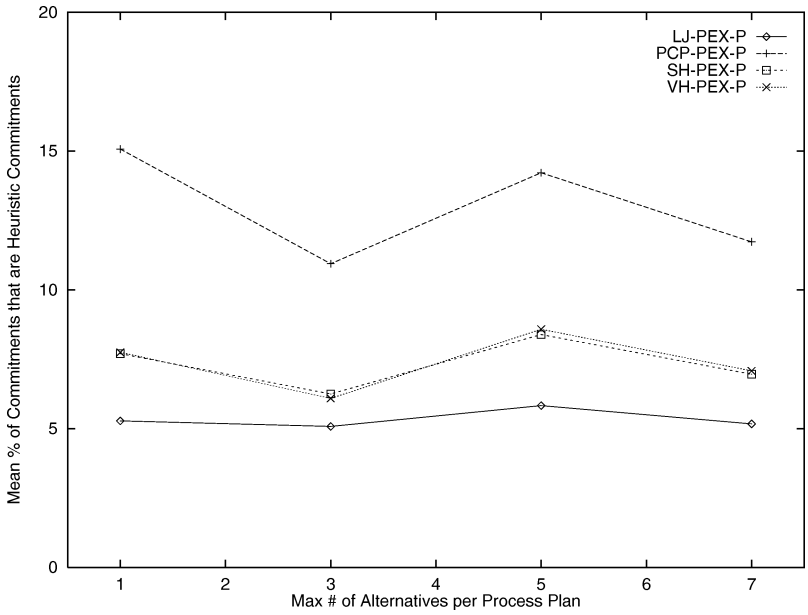


Fig. 16. The mean percentage of commitments in each problem set made by the heuristic commitment technique for Experiment 1.

However, the PCP-PEX heuristic, because it does not use the expected PEX values, does not benefit from the extra information that they represent.

An additional, and not incompatible, explanation for the PCP-PEX results vis-a-vis PEX-edge-finding is that PCP-PEX commitments tend to result in less propagation. In the job shop problems in [10], it was observed that a relatively large percentage of the PCP commitments were heuristic commitments as compared to those for SumHeight. It was argued that this arose from the fact that SumHeight tended to make a commitment where many activities were competing for a resource and time point while PCP simply looked for the most tightly constrained pair of activities. The subsequent propagation from a heuristic commitment therefore would tend to be higher when SumHeight is used. The same pattern can be seen here as shown in Fig. 16. Using the results from Experiment 1, we see that SH-PEX-P and VH-PEX-P make a significantly smaller percentage of heuristic commitments than PCP-PEX-P. While a PCP-PEX-P incurs the computational cost, the benefits are not apparent because the heuristic commitments do not result in significant propagation.

10. Related and future work

Representation and reasoning about alternative activities opens a number of areas of future research. While we have shown that the use of expected PEX values can lead to superior scheduling performance, there are a number of issues surrounding the scaling of the PEX approach that need to be addressed. In particular, the explicit representation of

each possible alternative activity results in scaling issues [19]. Investigation of methods to reduce the alternative activities that actually need to be represented forms important future tasks. Given the positive impact of the PEX-edge-finding propagator, techniques to reduce its average and worst-case time-complexity will also contribute to overall scheduling performance and help to combat scaling issues.

The PEX approach to alternative activities scheduling problems can be viewed as a bottom-up approach. Alternative activities are fully represented in the scheduling problem and choosing alternatives is fully integrated into the scheduling algorithm. In contrast, a top-down approach to scheduling with alternatives can be seen in number of places in the AI research literature [19,27,38,43,44]. In such systems, high-level and, in some cases, domain dependent, knowledge and heuristics are brought to bear on the problem of pruning alternatives before the actual scheduling takes place. While these techniques avoid many of the scaling problems of the PEX approach, the heuristics are not informed by as much detailed information as the PEX heuristics. As a result, the quality of the heuristic decisions and overall solution tends to be inferior to the PEX solution in those problems where the PEX approach is not overwhelmed by the size of the required representation [19]. Obviously, there is a trade-off in terms of the effort required to represent and reason about the detailed alternative information, and the quality of the heuristic decisions that can be made. Further research is required to better understand this trade-off and, perhaps, to find ways to combine the top-down approach with the bottom-up approach.

Reasoning about activity alternatives either with a PEX-based formulation or with a more top-down approach blurs the distinction between scheduling and planning. One of the major assumptions about the alternative scheduling problems addressed in this paper is that all activities are known before scheduling begins. Imagine, however, a problem with demands for a number of finished goods and a number of process plans with which each can be produced. Each finished good process plan may require a number of other inventories to be produced and these inventories, in turn, have alternative process plans which themselves require further inventories to be executed. This cascade of process plans terminates eventually with raw material supply events. The problem now begins to look more like planning (with scheduling constraints) than scheduling. A set of goals must be achieved; however, actions to achieve goals potentially conflict with one another and may introduce further sub-goals. The combination of scheduling and planning techniques to address such problems has been begun in work such as [27,38,39]; however, much remains to be investigated.

In this paper, we made the assumption that the expected PEX values are evenly distributed among the paths between a pair of XorNodes. This is a “low knowledge” assumption: we assume we have no other information with which we can bias the expected PEX values. In an application, we may have a preference for one alternative process plan over another. Given such a preference, the overall problem then begins to look more like a fuzzy or valued constraint satisfaction problem [20,34,40]. It would seem natural to incorporate these preferences by biasing the expected PEX values and, therefore, the individual texture curves. One difficulty with this approach, however, is the semantics of the individual demand curves. We have been interpreting these curves as a representation of the probability that an activity will demand a resource at a time point. It is not clear that this

interpretation is well-founded if the individual curves are biased by preferences. At the very least, the heuristic commitment technique used here does not extend to biased curves. The heuristic chooses, if possible, *not* to execute the activity with highest individual demand for a critical time point and resource. If the individual demand is high because there is a high preference to execute the activity, choosing not to execute the activity is unlikely to quickly lead to high quality solutions.

Finally, it should be noted that the use of PEX variables to govern the existence of activities can be seen as an instance of a Conditional Constraint Satisfaction Problem (CCSP) [35] (originally called Dynamic Constraint Satisfaction Problem [30]). In such a problem, only "active" variables need be assigned a value in a solution and "activity constraints" define which variables become active depending on the values assigned to subsets of other variables. Given an initial set of active variables, the assignment of a value to a variable may cause new variables to become active and, therefore, to require a value in order to solve the problem. Such a problem formulation is very natural for applications such as configuration [30]. In the work presented here, the PEX variables and any activities not enclosed by XorNodes form the initial set of active variables. The only type of activity constraint is the one that links the value of the PEX variable with the other variables in an Activity: if the PEX variable is assigned to 1, all the other variables in that activity are active, while if assigned to 0, the other variables are inactive. The link between our work and CCSP suggests that a possible area for future work is in generalizing the techniques used here. To our knowledge no one has either attempted to extend advanced constraint propagation techniques to CCSPs (such as we have done for PEX-edge-finding) nor attempted to incorporate the probability that a variable will be active into heuristic search techniques for CCSPs.

11. Conclusion

In this paper, we introduced the notion of the probability of existence (PEX) of an activity and used it to expand constraint-directed scheduling representation and reasoning to account for alternative activities: an activity present in the original problem definition does not necessarily have to be scheduled to achieve a solution.

The modeling of PEX required extensions to the constraint representation of activities and of the temporal network. Algorithms for the propagation of expected PEX values are presented along with modifications to temporal propagation to account for the fact that an activity might not exist in a final solution. In addition, heuristic commitment techniques and two edge-finding propagators are extended to account for the presence of PEX variables.

Experimental results indicated that incorporating expected PEX values into the texture measurements upon which the heuristic commitment techniques are based results in significantly higher quality commitments and better overall search performance. Performance differences are especially large when there is a wide range of expected PEX values in a problem. Experimental results also validated the use of PEX-edge-finding which, in most cases, leads to significantly better overall search performance.

Our primary theoretical motivation for the work in this paper is the investigation of the problem structure hypothesis which states that as a problem becomes more

complex, understanding its structure becomes increasingly important for successful heuristic search [41]. The results in this paper support the problem structure hypothesis: as the scheduling problems are made more complex by the addition of alternative activities, representation of the alternative information and heuristic reasoning that takes into account that information results in superior overall problem solving performance.

Acknowledgements

This research was performed while the first author was a Ph.D. student at 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 and Angela Glover for discussion of and comments on previous versions of this paper.

References

[1] P. Baptiste, C. Le Pape, Disjunctive constraints for manufacturing scheduling: Principles and extensions, in: Proc. 3rd International Conference on Computer Integrated Manufacturing, 1995.
[2] P. Baptiste, C. Le Pape, Edge-finding constraint propagation algorithms for disjunctive and cumulative scheduling, in: Proc. 15th Workshop of the UK Planning and Scheduling Special Interest Group, 1996. Available from http://www.hds.utc.fr/ baptiste/.
[3] P. Baptiste, C. Le Pape, W. Nuijten, Incorporating efficient operations research algorithms in constraint-based scheduling, in: Proc. 1st Joint Workshop on Artificial Intelligence and Operations Research, June 1995. Workshop proceedings available on world wide web from http://www.cirl.uoregon.edu/aior/.
[4] J.C. Beck, Texture measurements as a basis for heuristic commitment techniques in constraint-directed scheduling, Ph.D. Thesis, University of Toronto, 1999.
[5] J.C. Beck, A.J. Davenport, E.D. Davis, M.S. Fox, The ODO project: Toward a unified basis for constraint-directed scheduling, J. Scheduling 1 (2) (1998) 89–125.
[6] J.C. Beck, A.J. Davenport, C. Le Pape, Introduction to the special issue on industrial constraint-directed scheduling, CONSTRAINTS, in press.
[7] J.C. Beck, A.J. Davenport, E.M. Sitarski, M.S. Fox, Beyond contention: Extending texture-based scheduling heuristics, in: Proc. AAAI-97, Providence, RI, AAAI Press, Menlo Park, CA, 1997.
[8] J.C. Beck, A.J. Davenport, E.M. Sitarski, M.S. Fox, Texture-based heuristics for scheduling revisited, in: Proc. AAAI-97, Providence, RI, AAAI Press, Menlo Park, CA, 1997.
[9] J.C. Beck, K. Jackson, Constrainedness and the phase transition in job shop scheduling, Technical Report, School of Computing Science, Simon Fraser University, Burnaby, BC, 1997.
[10] J.C. Beck, M.S. Fox, Dynamic problem structure analysis as as basis for constraint-directed scheduling heuristics, Artificial Intelligence 117 (2000) 31–81.
[11] P. Brucker, O. Thiele, A branch & bound method for the general-shop problems with sequence dependent set-up times, OR Spektrum 18 (1996) 145–161.
[12] J. Carlier, E. Pinson, An algorithm for solving the job-shop problem, Management Sci. 35 (2) (1989) 164–176.
[13] J. Carlier, E. Pinson, Adjustment of heads and tails for the job-shop problem, European J. Oper. Res. 78 (1994) 146–161.
[14] Carnegie Group Inc., Knowledge-based logistics planning system, Internal Documentation, 1994.
[15] Y. Caseau, F. Laburthe, Improved CLP scheduling with task intervals, in: Proc. 11th International Conference on Logic Programming, MIT Press, Cambridge, MA, 1994.

[16] Y. Caseau, F. Laburthe, Improving branch and bound for jobshop scheduling with constraint propagation, in: Proc. 8th Franco–Japanese Conference CCS-95, 1995.
[17] C.C. Cheng, S.F. Smith, Applying constraint satisfaction techniques to job shop scheduling, Ann. Oper. Res. (Special Volume on Scheduling: Theory and Practice) 70 (1997) 327–378.
[18] P.R. Cohen, Empirical Methods for Artificial Intelligence, MIT Press, Cambridge, MA, 1995.
[19] A.J. Davenport, J.C. Beck, An investigation into two approaches for resource allocation and scheduling, in: INFORMS, 1999.
[20] D. Dubois, H. Fargier, H. Prade, The use of fuzzy constraints in job-shop scheduling, in: Proc. IJCAI-93, Chambéry, France, 1993.
[21] M.S. Fox, Constraint-directed search: A case study of job-shop scheduling, Ph.D. Thesis, Carnegie Mellon University, Intelligent Systems Laboratory, The Robotics Institute, Pittsburgh, PA, 1983. CMU-RI-TR-85-7.
[22] M.S. Fox, N. Sadeh, C. Baykan, Constrained heuristic search, in: Proc. IJCAI-89, Detroit, MI, 1989, pp. 309–316.
[23] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, New York, 1979.
[24] I.P. Gent, E. MacIntyre, P. Prosser, T. Walsh, The constrainedness of search, in: Proc. AAAI-96, Portland, OR, Vol. 1, 1996, pp. 246–252.
[25] I.P. Gent, T. Walsh, The hardest random SAT problems, in: B. Nebel, L. Dreschler-Fischer (Eds.), Proc. KI-94: Advances in Artificial Intelligence, 18th German Annual Conference on Artificial Intelligence, Springer, Berlin, 1994, pp. 355–366.
[26] D.W. Hildum, Flexibility in a knowledge-based system for solving dynamic resource-constrained scheduling problems, Ph.D. Thesis, Department of Computer Science, University of Massachusetts, Amherst, MA, 1994. UMass CMPSCI TR 94-77.
[27] A. Kott, V. Saks, A multi-decompositional approach to integration of planning and scheduling—An applied perspective, in: Proc. Workshop on Integrating Planning, Scheduling and Execution in Dynamic and Uncertain Environments, Pittsburgh, PA, June 1998.
[28] C. Le Pape, Using a constraint-based scheduling library to solve a specific scheduling problem, in: Proc. AAAI-SIGMAN Workshop on Artificial Intelligence Approaches to Modelling and Scheduling Manufacturing Processes, 1994.
[29] O. Lhomme, Consistency techniques for numeric CSPs, in: Proc. IJCAI-93, Chambéry, France, Vol. 1, 1993, pp. 232–238.
[30] S. Mittal, B. Falkenhainer, Dynamic constraint satisfaction problems, in: Proc. AAAI-90, Boston, MA, 1990, pp. 25–32.
[31] W.P.M. Nuijten, Time and resource constrained scheduling: A constraint satisfaction approach, Ph.D. Thesis, Department of Mathematics and Computing Science, Eindhoven University of Technology, 1994.
[32] W.P.M. Nuijten, E.H.L. Aarts, D.A.A. van Arp Taalman Kip, K.M. van Hee, Randomized constraint satisfaction for job shop scheduling, in: Proc. IJCAI-93, Chambéry, France, 1993, pp. 251–262.
[33] F.S. Roberts, Applied Combinatorics, Prentice Hall, Englewood Cliffs, NJ, 1984.
[34] Z. Ruttkay, Fuzzy constraint satisfaction, in: Proc. 3rd IEEE Conference on Fuzzy Systems, 1994, pp. 1263–1268.
[35] M. Sabin, E.C. Freuder, Detecting and resolving inconsistency and redundancy in conditional constraint satisfaction problems, in: Proc. CP-98 Workshop on Constraint Reformulation, 1998.
[36] N. Sadeh, Lookahead techniques for micro-opportunistic job-shop scheduling, Ph.D. Thesis, Carnegie-Mellon University, Pittsburgh, PA, 1991. CMU-CS-91-102.
[37] N. Sadeh, M.S. Fox, Variable and value ordering heuristics for the job shop scheduling constraint satisfaction problem, Artificial Intelligence 86 (1) (1996) 1–41.
[38] N.M. Sadeh, D.W. Hildum, T.J. Laliberty, J. McA'Nulty, D. Kjenstad, A. Tseng, A blackboard architecture for integrating process planning and production scheduling, Concurrent Engineering: Research and Applications 6 (2) (1998).
[39] V. Saks, Distribution planner overview, Technical Report, Carnegie Group, Inc., Pittsburgh, PA, 1992.
[40] T. Schiex, H. Fargier, G. Verfaillie, Valued constraint satisfaction problems: Hard and easy problems, in: Proc. IJCAI-95, Montreal, Quebec, 1995.
[41] H.A. Simon, The structure of ill-structured problems, Artificial Intelligence 4 (1973) 181–200.

[42] S.F. Smith, C.C. Cheng, Slack-based heuristics for constraint satisfaction scheduling, in: Proc. AAAI-93, Washington, DC, 1993, pp. 139–144.
[43] T. Wagner, A. Garvey, V. Lesser, Design-to-criteria scheduling: Managing complexity through goal-directed satisficing, in: Proc. AAAI-97, Providence, RI, 1997.
[44] T. Wagner, A. Garvey, V. Lesser, Criteria directed task scheduling, Internat. J. Approx. Reason. 19 (1998) 91–118.