ECAI-90

Why Is Scheduling Difficult? A CSP Perspective

Mark S. Fox & Norman Sadeh

Center for Integrated Manufacturing Decision Systems
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

msf@cs.cmu.edu & sadeh@cs.cmu.edu

Abstract

1

Scheduling differs from planning in that it assigns resources over time, and in the presence of constraints, to activities selected by a planner. This paper formulates the scheduling problem as an N-Castle CSP problem so that the relevance of CSP problem solving techniques can be determined. Problems with this formulation arise due to scheduling's infinite temporal domain, it being an optimization problem, and the need for constraint relaxation to find feasible solutions. Alternative heuristics for variable and value ordering are described based on measures of contention, reliance and survivability.

1. Introduction

The purpose of this paper is to acquaint the reader with the scheduling problem, show why it is a difficult problem to solve, and therefore worthy of our attention. Since the 1960s, the planning problem has captured the interest of many AI researchers. *Planning selects and sequences activities such that they achieve one or more goals and satisfy a set of domain constraints.* But it was only in the early 1980s that schedulir came under serious scrutiny, and moı recently has garnered the attention of a si nificant minority of researchers, primari in the domains of manufacturing an space[2]. *Scheduling selects among alte native plans, and assigns resources an times for each activity so that they obey tl temporal restrictions of activities and tl capacity limitations of a set of share resources.* It follows from these definitior that scheduling can be viewed as subclass planning, with the focus being on the alloc tion of resources over time. The recency the field's focus on scheduling is somewh odd, given that one of the earlier papers c planning explicitly pointed out the proble of allocating resources over time [Simon 72 On the other hand, the sterile world blocks never forced the issues that arise scheduling; it took a return to the "re world" for these issues to reappear.

From a computational complexity perspe tive, we know that both planning ar scheduling are difficult problems; they ha been proven to be NP-Hard [Chapma 87, Garey & Johnson 79]. But mo problems are NP, so that does not provi

[1]This research has been supported, in part, by the Defense Advance Projects Agency under contract #F30602-88-C-0001, and in part, by grants from Digital Equipment Corporation.

[2]There is even a conference on the topic, "Expe Systems in Production Operations ar Management", and workshops on the topic ha been held at AAAI.

insight into their solution. It is the intent of this paper to provide a different perspective on the the scheduling problem so that the methods being developed in AI can be applied more easily.

In earlier work, we have shown that one can view scheduling as a constraint optimization problem [Fox 83, Fox 86][3] This earlier work solved the scheduling problem by using constraints to heuristically direct search in the problem space [Fox & Smith 84, Ow & Smith 88, Fox 90]. The approach was synthetic in that it incrementally constructed a subset of partial schedules until one was found to be acceptable. Recently, a reductionist approach to scheduling, based on Constraint Satisfaction (CSP) techniques, has been explored. Techniques for constructing satisficing schedules [Eleby et al. 88, Keng et al. 88, Keng & Yun 89], and optimizing schedules [Sadeh & Fox 88, Sadeh & Fox 89, Fox et al. 89] have been demonstrated.

Viewing the scheduling problem from a CSP perspective can be useful. CSP is one of the few areas of AI where significant amounts of problem classification and complexity analysis has co-occurred [Mackworth 77, Haralick & Elliott 80, Freuder 82, Nudel 83, Purdom 83, Davis 87, Dechter & Pearl 87, Nadel 89]. Consequently, by reducing the scheduling problem to CSP, we can apply these results. On the other hand, as will be shown later, the scheduling problem extends beyond the current capabilities of CSP, providing for the carry over of methods from the scheduling domain.

In the following the complexity of the scheduling problem is explored through a series of factory scheduling problems. For each problem, an equivalent CSP formulation is provided. Finally, the difficulty of scheduling is analyzed and approaches to solving them are briefly described.

[3]This view is not unique; Operations Research has also formulated scheduling as an optimization problem and has applied various mathematical programming techniques, such as integer programming. See [Fox 91] for a comparison of AI and OR approaches.

2. Constraint Satisfaction Perpsective of Scheduling

CSP takes a reductionist approach to problem solving [Simon 83]; a super set of solutions are successively reduced to a solution set. A constraint satisfaction problem is defined by a set of variables $V=\{v_1,v_2,...,v_m\}$, each having a corresponding domain $D=\{d_1,d_2,...,d_m\}$, and a set of constraints $C=\{c_1,c_2,...,c_n\}$. A variable's domain d_i can be infinite, for example in the temporal domain, but is usually discrete and small. A constraint c_j, is an m-tuple that specifies a consistent assignment to the variables that it constrains, i.e., $c_i \subseteq d_1 \times d_2 \times \cdots \times d_m$. The process of solving a CSP is comprised of the following steps:

1. select a variable for instantiation,
2. select a value to assign to the variable, and then
3. determine whether the assignment is consistent with all the constraints. If not, then backtrack, otherwise iterate.

Research has focused on heuristics for selecting variables and values.

In the reminder of this section a series of factory scheduling problems are described and their CSP analog is formulated.

2.1. Single Resource Scheduling

The simplistic factory that one could imagine scheduling contains a single machine and produces a single product that requires a single operation. The scheduling goal is to assign each order for a product to an available time slot on the machine.

Equivalently, this can be stated more generally as a resource allocation problem where a single, indivisible resource, is to be allocated over time to n activities, but at any time to at most one activity. The activities are unrelated (i.e., no precedence relations among them) and are of equal duration.

From a CSP perspective, this is equivalent to what I call the ***N-Castle Problem.*** Given an n x n chess board, the problem is to place n castles such that they do not interfere according to chess rules. Unlike the N-Queens problem, more than one castle may occupy the same diagonal, but not the same row and column. Each castle, which occupies a separate row, corresponds to a separate activity and each column of the chess board corresponds to a unique, equal duration time slot that the activity can use the single resource.

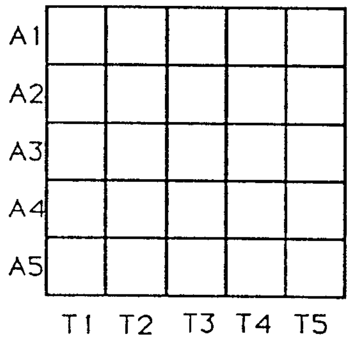

Figure 2-1: N-Castle Problem

More formally, given n activities A_i with domains $A_i \in \{1, \ldots, n\}$, assign a value to each subject to the following constraints:

$\forall ij\ [(i \neq j) \supset (A_i \neq A_j)]$
No two distinct activities may occupy the same column.

The constraint graph, whose nodes are activities, is completely connected by inequality constraints that assure that no activities occupy the same column/time slot. This problem is simpler than the N-Queens, and can be solved in polynomial time [Garey & Johnson 79].

2.2. Scheduling with Due Dates

To bring some realism to the scheduling problem, we impose the constraint that each activity A_i must be completed before a "due date" d_i. Each activity's due date is independent of the the due dates of other activities.[4] This is the same as "mutilating" the N-Castle chess board by removing squares at the end of each row. Assuming that each column is numbered from 1 to n, then if an activity A_i is due on date 5, then squares 6 through n in the activity's corresponding row i are unavailable for placing the castle.

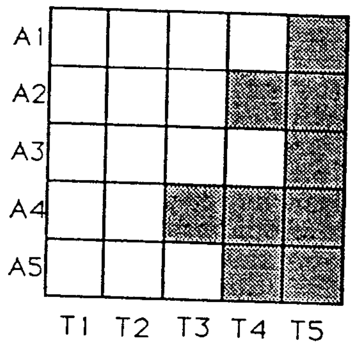

Figure 2-2: Mutilated N-Castle Problem

More formally, given n activities A_i with domains $A_i \in \{1, \ldots, n\}$, and due dates d_i. Assign a value to each subject to the following constraints

[4]Depending on how close the due dates are to each other, there may not exist sufficient time slots for all activities to be performed before or on their due dates.

$\forall ij\ [(i \neq j) \supset (A_i \neq A_j)]$
No two distinct activities may occupy the same column.

$\forall i\ [A_i \leq d_i]$
An activity must end on or before its due date.

The additional due date constraints serve to reduce the domain of each activity variable prior to any search being performed, thus simplifying the problem.

2.3. Activities With Precedence

Consider the same factory, but now each product is produced according to a process plan. A process plan defines a sequence of operations (or activities) that must be performed in the order specified.

More generally, the single resource scheduling problem is now further complicated by imposing precedence among activities. That is, for each activity, there may be zero or more activities that must be performed before it.

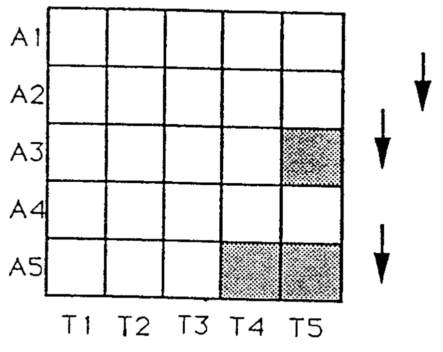

Figure 2-3: Mutilated N-Castle with Precedence

Formally, the Mutilated N-Castle problem with precedence is defined as given n activities A_i:

- with domains $A_i \in \{1, \ldots, n\}$,
- due dates $d_i \in \{1, \ldots, n\}$, and
- a precedence matrix P_{ij} where $P_{ij}=1$ if A_i must precede A_j,

assign a value to each subject to the following constraints:

$\forall ij\ [(i \neq j) \supset (A_i \neq A_j)]$
No two distinct activities may occupy the same column.

$\forall i\ [A_i \leq d_i]$
An activity must end on or before its due date.

$\forall ij\ [(P_{ij}=1) \supset (A_i < A_j)]$
Activities with precedence must be sequenced.

Initially, the constraint graph had all of the activity nodes connected together with inequalities, denoting that no castle/activity may occupy the same position. Precedence adds another layer of inter-activity constraints, as denoted by the P_{ij} precedence matrix.

2.4. Multiple Alternative Resources

To make the factory a little more realistic, more resources can be added. Now, each activity A_i may choose one of m resources to use, thus increasing the complexity of the task.

From a CSP perspective, the N-Castle problem can be further refined by extending the n x n chess board into a third dimension, each plane representing an alternative resource (figure 2-4). No two castles may occupy the same column within the same plane.

More formally, given n activities A_i and m alternative resources to choose from, with:

- activities having domains $A_i \in \{<T_i,R_i> \mid 1 \leq T_i \leq n,\ 1 \leq R_i \leq m\}$ where T_i is the time or column position of the activity and R_i is the resource or plane selected,
- due dates $d_i \in \{1, \ldots, n\}$, and

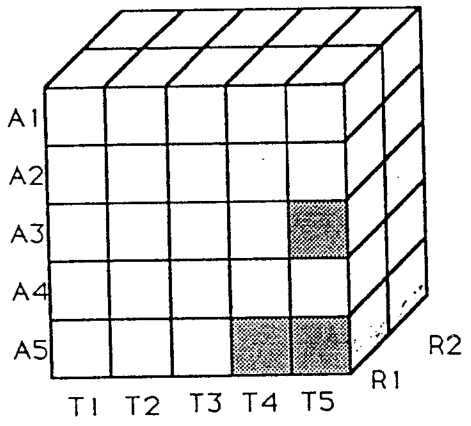

Figure 2-4: 3D Mutilated N-Castle with Precedence Problem

- a precedence matrix P_{ij} where $P_{ij}=1$ if A_i precedes A_j,

assign a value to each subject to the following constraints:

$\forall i\ [T_i \leq d_i]$
An activity must end on or before its due date.

$\forall ij\ [(P_{ij}=1) \supset (T_i<T_j)]$
If precedence exists among activities i and j, then activity i must be assigned a position before activity j.

$\forall ij\ [((i \neq j) \wedge (R_i=R_j)) \supset (T_i \neq T_j)]$
No two distinct activities using the same resource may occupy the same time time slot.

Nodes in the constraint graph are now 2-tuples; significantly enlarging the space of alternative potential solutions.

2.5. Interfering Activities

The formulation provided so far is general enough to specify a flow shop (and actually a job shop) where each product goes through the same sequence of activities. Now consider an assembly line, a type of flow shop, where a small number of product types are sequenced to be released to the line. Each type of product has a different set of component parts, or "options". The assembly line is balanced so that no two products of the same type can be within b_i positions of each other. Otherwise the line would have to be temporally halted so that more time is available to complete the activities.

This is equivalent to our latest version of the N-Castle problem, where castles can be of different colors, and the lines of interference are defined as regions. That is, there cannot be castles of the same color within a region centered at a castle, in the same plane.

Formally, given n activities A_i and m alternative resources to choose from, with:

- activities having domains $A_i \in \{<T_i,R_i> \mid 1 \leq T_i \leq n,\ 1 \leq R_i \leq m\}$ where T_i is the time or column position of the activity and R_i is the resource or plane selected,
- due dates $d_i \in \{1, \ldots, n\}$,
- a precedence matrix P_{ij} where $P_{ij}=1$ if A_i precedes A_j,
- an activity type $AT_i \in \{1, \ldots, o\}$, that is, it is one of o types, and
- a neighborhood around a castle of type t of size $B(t)$,

assign a value to each subject to the following constraints:

$\forall i\ [T_i \le d_i]$
An activity must end on or before its due date.

$\forall ij\ [(P_{ij}=1) \supset (T_i < T_j)]$
If precedence exists among activities i and j, then activity i must be assigned a position before activity j.

$\forall ij\ [((i \ne j) \wedge (R_i = R_j)) \supset (T_i \ne T_j)]$
No two distinct activities using the same resource may occupy the same time slot.

$\forall ij\ [((AT_i = AT_j) \wedge (R_i = R_j)) \supset (|T_i - T_j| > B(AT_i))]$
If activities i and j are of the same type and use the same resource, the "distance" between them should be greater than specified by its type.

The delineation of regions around types of activities complicates the assignment problem. Poor initial assignments of activities may not leave open regions large enough for subsequent activities of the same type. Therefore assigning activities with the largest regions first would appear to be a good idea, and the smaller activities could be inserted among them.

2.6. Non-Uniform Durations

Factories produce more than one product. Each product may have a different sequence of activities, the number of activities in a sequence may vary, and the duration of each activity may vary. It is the last point that complicates the scheduling problem further.

To accomm'<odate this last point, the N-Castle problem is modified so that a castle may occupy more than one square, and the number of squares each castle occupies may differ.

Formally, given n activities A_i and m alternative resources to choose from, with

- activities having domains $A_i \in \{<T_i,R_i> \mid 1 \le T_i \le n,\ 1 \le R_i \le m\}$ where T_i is the time or column position of the activity and R_i is the resource or plane selected,
- due dates $d_i \in \{1, \ldots, n\}$,
- a precedence matrix P_{ij} where $P_{ij}=1$ if A_i precedes A_j,
- an activity type $AT_i \in \{1, \ldots, o\}$, that is, it is one of o types, and
- the number of squares occupied by the ith activity is defined by S_i,

assign a value to each subject to the following constraints:

$\forall i\ [(T_i + S_i - 1) \le d_i]$
An activity must end on or before its due date.

$\forall ij\ [(P_{ij}=1) \supset ((T_i + S_i - 1) < T_j)]$
If the ith activity must precede the jth activity, then the ith activity must end before the jth begins.

$\forall ij\ [(R_i = R_j) \supset (((T_j + S_j - 1) < T_i) \vee (T_j > (T_i + S_i - 1)))]$
If two activities use the same resource, then they must not overlap in time.

With non-uniform durations, choices of where to place an activity can have an enormous impact. If assignments do not leave gaps large enough for subsequent assignments, then significant amounts of backtracking can arise. One would think then that you would assign activities with the largest durations first[5].

2.7. Multiple Resources and Variable Durations

The scheduling problem can be generalized further to a *job shop without assemblies*, defined as follows:

- The factory produces two or more different products, where similar products are grouped into families.
- Each product requires one or more

[5]Operations research has studied a variety of heuristics, such as shorted processing time or earliest due date. Both are appropriate under different conditions [Baker 74].

operations to produce it, where the operations sequentially transform basic materials into the final product.

- The sequence of operations required to produce a product is defined to be its process plan. Process plans differ for each product.
- Each operation specifies one or more resources that are required during its performance.
- Durations for operations are specified apriori but may be contracted if orders for products in the same family follow each other on the same resource.
- Products are produced on demand, that is on the receipt of orders.
- There are multiple orders in production at any time. Contention usually exists for a subset of resources.
- Lead times for the delivery of orders can vary from zero days to multiples of the actual manufacturing lead time.

The job shop complicates the scheduling problem because (1) activities may use more than one resource, thereby increasing the degree of interference with other activities, and (2) the sequencing of activities at a specific resource may result in an expansion or contraction in the amount of time an operation will use one or more resources. Another way of viewing context dependent operation durations is that each operation can be viewed as a set of alternative operations, with different durations, and constraints that limit their selection based on the operation that precedes it at the same resource. That is, if the prior abuting operation is in the same family, then the operation with the reduced duration can be selected.

Formally, given n activities A_i and m alternative resources to choose from, with

- activities having domains $A_i \in \{<T_i,R_i,V_i> \mid 1 \leq T_i \leq n, R_i \in Powers$
  T_i is the time or column position of the activity, R_i is the set of resources selected, and V_i is the version of the activity whose choice is constrained by the activity that precedes it,
- $Version_{ij}$ specifying the version of a
  tivity j if preceded by activity i,
- due dates $d_i \in \{1, \ldots, n\}$,
- a precedence matrix P_{ij} where $P_{ij}=1$
  A_i precedes A_j,
- an activity type $AT_i \in \{1, \ldots, o\}$, tha
  is, it is one of o types, and
- the number of squares occupied by th
  i^{th} activity is defined by S_{iv} (note tha
  the amount depends on the version
  the activity chosen,

assign a value to each activity subject to th
following constraints:

$\forall i\ [(T_i+S_i-1) \leq d_i]$
An activity must end on or before its due date.

$\forall ij\ [(P_{ij}=1) \supset ((T_i+S_i-1)<T_j)]$
If the ith activity must precede the jth activity, then the ith activity must end before the jth begins.

$\forall ij\ [((R_i \cap R_j) \neq \emptyset) \supset$
$(((T_j+S_j-1)<T_i) \vee (T_j>(T_i+S_i-1)))]$
If two activities use the same resource, then they must not overlap in time.

$\forall j\, \exists i\ [Directly\text{–}Precedes(ij) \supset$
$(V_i=Version_{ij})]$
If there exists an activity i that directly precedes activity j, where Directly-Precedes is a predicate, then the activities version is specified by the matrix Version.

With activities requiring more than on
resource at a time, the degree of interferenc
among activities continues to increase
Secondly, with durations being variable, i
becomes more difficult to predict the impac
of an assignment on subsequent assign
ments.

The job shop can be complicated further b
extending process plans from chains t
graphs; any path through the process plan'
graph defines a legal means of producin
the product. A formal specification of thi

version of the scheduling problem is left to the reader!

3. Why Is Scheduling So Difficult?

3.1. Temporal Complexity

Schedules are constructed to span a *temporal horizon*, that is, detailed schedules are produced over some time interval. The length of the horizon depends upon the lead time with which resources, to be used in the production of the order, have to be planned and sourced. In some cases it may be weeks, and in other cases it may be years. Over the temporal horizon, schedules must describe activities to a particular *temporal granularity*, perhaps to a day, shift, hour or minute. The temporal granularity of a schedule depends upon the duration of the activities and the degree of uncertainty in the environment; that is, to what extent schedules can be followed due to stochastic events such a resource unavailability. Depending on the granularity the number of start times for each activity to choose from can be large.

Consider the simple scheduling problem with uniform durations and alternative resources. An upperbound on the number of solutions to this problem for the case of 100 activities, with 100 time slots, each having the choice of one of 100 resources is determined as follows. By restricting each activity/castle to a plane, each activity has at most 10^4 positions to choose from. Given that there are 10^2 activities, then the cross product of all these activities is 10^{400}.

For extended horizons or more precise granularities, the number of starting times is enormous. Consequently, the N-castle formulation is only an abstraction of the real problem. Experience has shown that a change is required to an interval representation of time. In systems such as ISIS [Smith 83] and Deviser [Vere 83] have used variants of Allen's temporal relations [Allen 83] to identify scheduling intervals.

3.2. Optimization

The CSP perspective is that any variable assignment that satisfies the set of constraints is equally acceptable. That is not the case in the scheduling domain. Due dates may be extended indefinitely into the future and there may exist many solutions to a scheduling problem, some which are better than others. Scheduling is really an optimization problem, the goal being to optimize criteria, such as:

- ***Lateness:*** Minimize the amount of time between when an activity is completed and its due date.
- ***Flow Time:*** Minimize the amount of time it takes for a product to complete its activities; that is, how much time the product spends being worked on in the factory.
- ***Cost:*** Minimize the amount of money spent on producing the product.

The consequence of this is that CSP definition is not sufficient to encompass the entire set of scheduling problems. Instead, a second class of problems, called Constrained Optimization Problems (COP) [Fox et al. 89] is defined similar to CSPs, but has, in addition, an ***objective function***, that provides a numerical prioritization of proposed solutions. How the objective function is used during search is an interesting problem (see [Sadeh & Fox 88] for one approach).

3.3. Feasibility and Relaxation

Up till now we have taken for granted that a feasible solution exists to the aforementioned variations of the N-Castle problem. This is not often the case in factory settings. Due to the costs involved, resources are not available at levels that are sufficient to satisfy the temporal requirements imposed by due date and precedence constraints. Consequently, there may not exist a feasible solution.

In the factory scheduling domain it is not acceptable to just recognize that there does not exist a solution. Rather, as good a solution as possible must be found, even if a subset of the constraints are not satisfied. The question then is what subset of constraints are to be relaxed and how. Often, within a domain there is a clear weighting of constraints, and in some cases relaxations are specifiable. Some should not be relaxed, such as capacity constraints, but others can be relaxed, such as cost or due dates. This information may be utilized in the process of relaxation. In ISIS, OPIS, and CORTES, relaxations and their utilities are defined explicitly as part of the constraint representation. A variety of techniques have been explored for deciding when and where to relax a constraint [Fox 90]. Alternatively, Operations Research would view a due date not as a constraint but as part of the objective function with the intent to minimize the difference between an activity's actual completion time and its due date. A third approach within the CSP framework randomly chooses constraints to remove from the problem [Freuder 89].

In order to address the issue of achieving feasibility via relaxation, the COP constraint graph has to be extended to include information, such as the following:

- Relative weightings of constraints.
- Explicit specifications of relaxations of constraints.
- Utilities of relaxations.

Such representations have arisen in the constraint directed scheduling techniques discussed above, and in relaxation labelling techniques [Zucker 76].

3.4. Variable and Value Ordering Heuristics

How would we solve the scheduling problem with a CSP approach? First, lets ignore the issues of optimization and relaxation.

Some preprocessing of the constraint network can be performed to reduce the domains of the activities. For example, the satisfaction of temporal constraints, such as due dates and precedence, will reduce the start times of activities to a set of intervals. But the main effort is in the variable and value selection.

The most widely accepted heuristic for selecting the next variable is to choose one with the smallest remaining domain [Bitner & Reingold 75]. It has been shown that this technique can greatly reduce search complexity [Purdom 83]. The question is whether it is sufficient for scheduling. It has been shown experimentally th simple heuristic is not sufficient [Sac If an activity has only a few start ti resources relative to other activitie backtracking, but as you will see lat is not always the case. But this is not sue. For most activities, there will be times and resources available, thus domain 's will be approximately eq size. The issue is how to distingui tween a large number of what appea indistinguishable activities? Instea activities that should be scheduled fir the activities with a large proportion o possible reservations that are expect become unavailable if other activitie scheduled earlier.

For value selection, the concept of goodness has been proposed [Dechte Pearl 87]. That is, choose a value that ticipates in the largest number of solut Since generating all possible solutions i feasible, Dechter and Pearl proposed usi tree-like relaxation of the problem to proximate value goodness. For the sche ing problem we have found that there i good tree-like relaxation, other relaxat have been explored and are described in next section.

4. The CORTES Approach

CORTES is a factory scheduling sys [Fox & Sycara 90] that solves resource location COP's using Constrained Heuri Search (CHS) [Fox et al. 89]. Underly CORTES's approach to solving resource location COPs is a generalization and f malization of what is known in the sched ing literature as bottleneck schedulin That is, in order to optimize a schedule, fi optimize the scheduling of bottlene resources.

For a particular scheduling proble CORTES constructs a constraint graph th differs from a CSP constraint graph in number of ways. First, if determining t bottleneck resources is critical to making scheduling decision, then a typical co straint graph with activities as nodes hide this information. In order to determine th degree of *contention* for a resource, a redu dant version of original constraint graph i needed. In this constraint graph there are two types of nodes, activity and resource nodes. Rather than burying the resource in the activity node, the demands for each resources are represented explicitly by capacity demand constraints linking activities to resources. A second change to the constraint graph is to include relaxations and their utilities. Rather than specifying a single due date, a set of alternative dates with associated utilities are specified.

Scheduling using this constraint graph follows the following process:

1. Temporal preferences such as reducing tardiness and inventory are propagated across the temporal constraints [Sadeh & Fox 88, Sadeh & Fox 90].
2. The resulting preferences are further propagating across the capacity demand constraints while accounting for possible resource preferences.
3. The resulting demand profiles provide measures for resource contention as a function of time.
4. Highly contended resource/time intervals help identify critical activities. More specifically, the activity whose demand relies most on the contended resource/time intervals is selected to be scheduled next.
5. A reservation (start time and a set of resources) is selected for this activity by accounting for its direct contribution to the objective function (local preferences) and its liklihood to survive the contention of other unscheduled activities.

The contention metric is a heuristic approximation of the more general notion of constraint tightness. Constraint tightness manifests itself in the scheduling problem around the available resource capacity over time, i.e., the capacity constraints. By accounting for contention in an activity's criticality, we obtain a more precise indicator of the importance of selecting an activity than provided by previous variable ordering heuristics.

A detailed description of this approach can be found in [Sadeh & Fox 90].

5. Conclusion

Constraint Satisfaction research has made great strides in understanding the power of heuristics in solving large systems of constraints. But the continued focus on N-Queens and satisfiability problems has led CSP research to ignore additional problem characteristics and rich constraint structure that arise in specific classes problems. Our experience in the scheduling domain has demonstrated both the relevance and limitations of CSP techniques. What makes scheduling a difficult problem to solve from a CSP perspective is:

1. Scheduling is an optimization problem in a very large combinatorial space. Therefore a good solution must be found as quickly as possible.
2. In most scheduling problems there exists a plethora of constraints. It is often the case that the problem is infeasible, requiring that one or more constraints be relaxed in order to find a solution.
3. CSP representations have insufficiently represented the role of domain values. That is, in scheduling problems, the degree to which resources are contended for and activities rely upon them are important factors in making variable and value ordering decisions.

Our work on CORTES describes how many of these shortcoming can be addressed in solving resource constrained optimization problems.

References

[Allen 83] Allen, J.F.
Maintaining Knowledge about Temporal Intervals.
Communications of the ACM 26(11):832-843, 1983.

[Baker 74] Baker, K.R. *Introduction to Sequencing and Scheduling.* John Wiley & Sons, 1974.

[Bitner & Reingold 75] Bitner, J.R., and Reingold, E.M. Backtrack Programming Techniques. *Communications of the ACM* 18:651-655, 1975.

[Chapman 87] Chapman, D. Planning for Conjunctive Goals. *Artificial Intelligence* 32:333-377, 1987.

[Davis 87] Davis, E. Constraint Propagation with Interval Labels. *Artificial Intelligence* 32:281-331, 1987.

[Dechter & Pearl 87] Dechter, R. and Pearl, J. Network-based Heuristics for Constraint-Satisfaction Problems. *Artificial Intelligence* 34(1):1-38, 1987.

[Eleby et al. 88] Eleby, P., Fargher, H.E., and Addis, T.R. Reactive Constraint-Based Job-Shop Scheduling. *Expert Systems and Intelligent Manufacturing.* Elesevier Science Publishing Co., 1988, pages 1-10.

[Fox 83] Fox, M.S. *Constraint-Directed Search: A Case Stu of Job-Shop Scheduling.* Technical Report, Carnegie-Mellon University, 1983. CMU-RI-TR-85-7, Intelligent Systems Laboratory, The Robotics Institute, Pittsburgh,PA.

[Fox 86] Fox, M.S. Observations on the Rol of Constraints in Problem Solving. In *Annual Conference of the Canadian Societ for the Computationa Studies of Intelligenc* University of Quebec Press, 1986.

[Fox 90] Fox, M.S. Constraint Guided Scheduling: A Short History of Schedulin Research at CMU. *Computers and Industry* 1990. To Appear.

[Fox 91] Fox, M.S. Reflections on the Relationship of AI to OR. *Interfaces* , 1991. To Appear.

[Fox & Smith 84] Fox, M.S., and Smith, S. ISIS: A Knoweldge-Based System for Factory Scheduling. *International Journal of Expert Systems* 1(1):25-49, 1984.

[Fox & Sycara 90] Fox, M.S. and Sycara, K.
Overview of CORTES: A Constraint Based Approach to Production Planning, Scheduling and Control.
In *Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management.* May, 1990.
Submitted for publication.

[Fox et al. 89] Fox, M.S., Sadeh, N., and Baykan, C.
Constrained Heuristic Search.
In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 309-316. Morgan Kaufmann Pub. Inc., 1989.

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

[Freuder 89] Freuder, E.C.
Partial Constraint Satisfaction.
In *Eleventh International Joint Conference on Artificial Intelligence*, pages 278-283. Morgan Kaufmann Publishers Inc., 1989.

[Garey & Johnson 79]
Garey, M.R., and Johnson, D.S.
Computers and Intractability: A Guide to the Theory of NP-Completeness.
Freeman and Co., 1979.

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

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

[Keng et al. 88] Keng, N.P., Yun, D.Y.Y., and Rossi, M.
Interaction-sensitive planning system for job-shop scheduling.
Expert Systems and Intelligent Manufacturing :57-69, 1988.

[Mackworth 77] Mackworth, A.K.
Consistency in Networks of Relations.
Artificial Intelligence 8(1):99-118, 1977.

[Nadel 89] Nadel, B.A.
Constraint Satisfaction Algorithms.
Computational Intelligence 5(4):188-244, 1989.

[Nudel 83] Nudel, B.A. Consistent Labelling Problems and their Algorithms: Expected-Complexities and Theory-Based Heuristics. *Artificial Intelligence* 21(1 & 2):135-178, 1983.

[Ow & Smith 88] Ow, P.S., and Smith, S.F. Viewing Scheduling as an Opportunistic Problem-Solving Process. *Annals of Operations Research* 12:85-108, 1988.

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

[Sadeh 89] Sadeh, N. *Lookahead Techniques for Activity-based Job-shop Scheduling.* Technical Report, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, 1989. Thesis Proposal.

[Sadeh & Fox 88] Sadeh, N., and Fox, M.S. *Preference Propagation in Temporal Constraints Graphs.* Technical Report, Intelligent Systems Laboratory ,The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, 1988. CMU-RI-TR-89-2.

[Sadeh & Fox 89] Sadeh, N. and Fox, M.S. Focus of Attention in an Activity-based Scheduler. In *Proceedings of the NASA Conference on Space Telerobotics.* 1989.

[Sadeh & Fox 90] Sadeh, N., and Fox, M.S. Focusing Attention in an Activity-based Job-Shop Scheduler. In *Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management.* May, 1990.

[Simon 72] Simon, H.A. On Reasoning About Actions. *Representation and Meaning: Experiments with Information Processing Systems.* Prentice-Hall, 1972, page 414-430.

[Simon 83] Simon, H.A. Search and Reasoning in Problem Solving. *Artificial Intelligence* 21:7-29, 1983.

[Smith 83] Smith, S.F. *Exploiting Temporal Knowledge to Organiz Constraints.* Technical Report, Robotic Institute, Carnegie Mellon University, Pittsburgh, PA 15213, 1983.

[Vere 83] Vere, S.
Planning in Time: Windows and Durations for Activities and Goals.
IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-5(3):246-267, 1983.

[Zucker 76] Zucker, S.W.
Relaxation Labelling and the Reduction of Local Ambiguities.
Pattern Recognition and Artificial Intelligence.
Academic Press, 1976.

[illegible]