Generating Feasible Schedules under Complex Metric Constraints *

Cheng-Chung Cheng
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
ccen@isl1.ri.cmu.edu

Stephen F. Smith
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
sfs@isl1.ri.cmu.edu

Abstract

In this paper, we consider the problem of finding feasible solutions to scheduling problems that are complicated by separation constraints on the execution of different operations. Following recent work in constraint-posting scheduling, we formulate this problem as one of establishing ordering relations between pairs of operations requiring synchronization of resource usage. This establishes contact with the recently proposed General Temporal Constraint Network (GTCN) model. Exploiting properties of the general GTCN solution procedure, we are able to directly generalize a high performance solution procedure previously developed for a much more restricted class of scheduling problems. Specifically, shortest path information in the underlying temporal constraint network is first used to establish dominance conditions for early pruning of infeasible solutions. Heuristics are then defined for variable/value ordering in the meta-CSP space of possible resolutions of the disjunctive constraints on resource usage. These heuristics are based on use of shortest path information as an estimation of decision flexibility. Experimental evaluation of the resulting heuristic procedure is carried out on a set of randomly generated problems drawn from a manufacturing scenario. Results indicate extremely effective problem solving performance in relation to both the original scheduling procedure that was adapted and the general GTCN solution procedure.

Introduction

Scheduling problems present an important and challenging class of constraint satisfaction problems (CSPs) that have received increasing attention in the literature (e.g., Keng & Yun 1989, Minton et al. 1992, Muscettola 1993, Sadeh & Fox 1990, Smith & Cheng 1993). The classic problem typically considered, which originates from the manufacturing domain and is referred to as the job shop scheduling problem, involves synchronizing the production of *n jobs* in a facility with *m resources*. The production of a given job requires the execution of a sequence of *operations*. Each operation has a specific processing time and its execution requires the exclusive use of a designated resource (i.e. resources have unit capacity). Each job has an associated ready time and a deadline, and its production must be accomplished within this interval. In the non-relaxable version of the problem, the objective is to determine a set of operation start times that satisfies all temporal and resource capacity constraints.

This problem bears similarity to the problems of interest in the related field of temporal reasoning. Historically there have been differences. Qualitative temporal reasoning (e.g., Allen 1983, Vilain & Kautz 1986) has been concerned with problems of establishing and verifying consistent relations between events, but faces difficulties in incorporating metric information such as durations, ready times and deadlines; Quantitative approaches (Dechter, Meiri, & Pearl 1991) have investigated problems with complex metric constraints but are unable to accommodate disjunctive resource capacity constraints. More recent work in integrating qualitative and quantitative temporal reasoning (Dean & McDermott 1987, Meiri 1991, Kautz & Ladkin 1991) has proposed models that encompass the full set of constraints of the scheduling problem summarized above. In fact, one of these models (Meiri 1991) corresponds directly to the model used in (Smith & Cheng 1993), where an efficient heuristic solution procedure for the job shop scheduling problem was developed.

By exploiting this connection and formulating the scheduling problem within Meiri's model, we gain insight into the development of high performance heuristic procedures for a much wider class of frequently encountered scheduling problems. The specific class of scheduling problems we address in this paper extends the classical job shop problem in several respects. First, the precedence constraints that define the operation sequences of jobs are extended to allow additional specification of metric "separation constraints", delineating a feasible interval within which the execution of related operations can be separated in time.

*The research reported in this paper was supported in part by the Advanced Research Projects Agency under contract F30602-90-C-0119, the National Aeronautics and Space Administration under contract NCC 2-531 and the CMU Robotics Institute.

Second, the assumption of fixed operation processing times is relaxed to instead allow specification of minimum and maximum bounds. Finally, temporal relationships are specifiable between operations in different job sequences, allowing synchronization constraints unrelated to resource usage. All of these extensions find direct application in many practical domains. In metal parts manufacturing, for example, if an operation heats metal for a subsequent shaping operation, then the shaping operation must be executed before the metal cools for the process to be productive. Alternatively, a chemical bath operation in chip manufacturing processes requires a certain amount of time to be productive, but damage to the part is typically only incurred if the time in the bath exceeds a larger, maximum amount of time. The development of procedures for solving this extended class of scheduling problems has received very little attention from either the Artificial Intelligence or Operations Research communities.

The remainder of the paper is organized as follows. We first summarize the general temporal constraint network (GTCN) proposed by Meiri. Next, we use this model to characterize our extended scheduling problem, and discuss the general GTCN solution procedure in this problem context. We then turn to development of an efficient heuristic procedure for scheduling under complex metric constraints. After specifying the procedure, an experimental analysis of its performance is presented.

General Temporal Constraint Networks

A general temporal constraint network T consists of a set of variables $\{X_1, ..., X_n\}$ with continuous domains, and a set of unary or binary constraints. Each variable represents a temporal object, either a time point or an interval, and a constraint C may be qualitative or metric.

A qualitative constraint C is represented by a disjunction $(X_i\ r_1\ X_j) \vee ... \vee (X_i\ r_k\ X_j)$, alternatively expressed as a relation set $X_i\ \{r_1, ..., r_k\}\ X_j$, where r_i represents a *basic qualitative constraint*. Three types of basic qualitative constraints are allowed: (1) interval-interval constraints (Allen 1983); (2) point-point constraints (Vilain & Kautz 1986); (3) point-interval or interval-point constraints (Ladkin & Maddux 1989).

A metric constraint C is represented by a set of intervals $\{I_1, ..., I_k\} = \{[a_1, b_1], ..., [a_k, b_k]\}$. Two types of metric constraints are specifiable. A unary constraint C_i on point X_i restricts X_i's domain to a given set of intervals, i.e. $(X_i \in I_1) \vee ... \vee (X_i \in I_k)$. A binary constraint C_{ij} between points X_i and X_j restricts the feasible values for the distance $X_j - X_i$, i.e., $(X_j - X_i \in I_1) \vee ... \vee (X_j - X_i \in I_k)$. A special time point X_0 can be introduced to represent the "origin". Since all times are relative to X_0, each unary constraint C_i can be treated as a binary constraint C_{0i}.

A general temporal constraint network is associated with a *directed constraint graph* G, where nodes represent variables, and a edge $i \rightarrow j$ indicates that a constraint C_{ij} between variables X_i and X_j is specified. We say a tuple $X = (x_1, ..., x_n)$ is a *solution* if X satisfies all qualitative and metric constraints. A network is consistent if there exists at least one solution.

J_1 - ready time: 4; deadline: 16

operation	resource	duration
O_1	R_1	[4,8]
O_2	R_2	[3,9]

J_2 - ready time: 2; deadline: 16

operation	resource	duration
O_3	R_1	[5,8]
O_4	R_2	[2,7]

separation constraints:

[3,10] between O_1 and O_2

Figure 1: A simple scheduling example

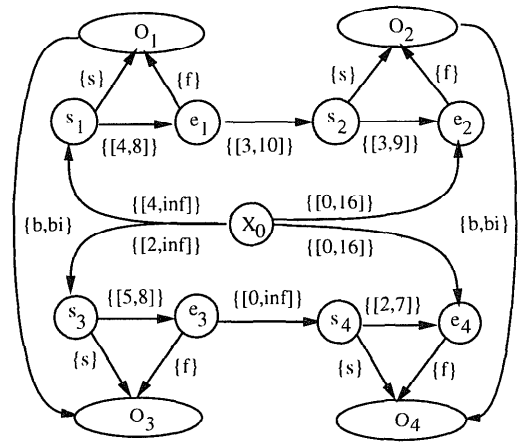

Figure 2: The constraint graph of the example

Modeling the Scheduling Problem

To illustrate how to model the scheduling problem using GTCN, consider a simple scenario involving two jobs J_1 and J_2. J_1 requires the operation sequence $O_1 \rightarrow O_2$, and J_2 requires the operation sequence $O_3 \rightarrow O_4$. Figure 1 gives the ready times and deadlines for J_1 and J_2, the resource required by each operation, and duration and separation constraints. Figure 2 shows the corresponding constraint graph for this problem. For expository purposes, we display an interval object by an oval and a point object by a circle.

In the graph, s_i and e_i represent the start and end points for operation O_i respectively, and X_0 is the origin, representing "time zero". For operations O_i and O_j, requiring use of the same resource, we model the *resource capacity constraint* as a relation set $\{b, bi\}$ between them, indicating that O_i should be sequenced *before* or *after* O_j without any overlap. For an exam-

Table 1: The formal specification of constraints

ready times:	$s_1 - X_0 \in [4, \infty]$	durations:	$e_1 - s_1 \in [4, 8]$
	$s_3 - X_0 \in [2, \infty]$		$e_2 - s_2 \in [3, 9]$
deadlines:	$e_2 - X_0 \in [0, 16]$		$e_3 - s_3 \in [5, 8]$
	$e_4 - X_0 \in [0, 16]$		$e_4 - s_4 \in [2, 7]$
capacity	$O_1\{b, bi\}O_3$	start &	$s_1\{s\}O_1; e_1\{e\}O_1$
constraints:	$O_2\{b, bi\}O_4$	end	$s_2\{s\}O_2; e_2\{e\}O_2$
separation	$s_2 - e_1 \in [3, 10]$	points:	$s_3\{s\}O_3; e_3\{e\}O_3$
constraints:	$s_4 - e_3 \in [0, \infty]$		$s_4\{s\}O_4; e_4\{e\}O_4$

ple of resource capacity constraint, see the constraint $\{b, bi\}$ between O_1 and O_3.

Job ready times and deadlines are represented by metric constraints between point X_0, and the corresponding start or end points. For example, the ready time 4 of job J_1 is represented by a metric constraint $\{[4, \infty]\}$ between point X_0 and J_1's start point s_1, indicating that the start time of J_1 should be greater than or equal to the ready time 4.

A separation constraint is represented by a metric constraint between the end and start points of the prescribed operations (e.g. the constraint $\{[3, 10]\}$ between e_1 and s_2). A simple precedence constraint (which is always assumed in the classical scheduling problem) is just a special case where the separation interval is unbounded (e.g., the constraint $\{[0, \infty]\}$ between e_3 and s_4). An operation's processing time is similarly represented by a metric constraint, in this case between its start and end points (e.g., the constraint $\{[4, 8]\}$ between s_1 and e_1). Table 1 shows the formal specification of all constraints graphically depicted in Figure 1(b).

Scheduling problems formulated in this manner represent a subclass of general temporal constraint problems where the only disjunctions present in the associated constraint graph are the $\{b, bi\}$ edges corresponding to the resource capacity constraints. All other edges have either one relation or a single interval. It is well known that this class of problems is strongly NP-complete (Garey & Johnson 1979), and heuristics are required for efficient solution.

A General Backtracking Procedure

By determining a feasible schedule, we mean to find a solution of the associated general temporal constraint network. A straightforward way of solving a GTCN has been suggested in (Meiri 1991). Let a *labeling* of a general temporal constraint network, T, be a selection of one relation from each qualitative constraint or one interval from each metric constraint. Since each basic qualitative constraint can be translated into at most four metric constraints (Kautz & Ladkin 1991), a labeling actually defines a Simple Temporal Problem (STP) network - a metric network whose arcs have only single intervals (Dechter et al. 1991). We can solve T by generating all possible labelings, solving each one of them, and combining the results. Specifically, T is consistent if and only if there exists a labeling whose associated STP is consistent. In our scheduling problem, most arcs are labeled by single relations or single intervals, except for those labeled by the relation set $\{b, bi\}$. To generate all labelings, we exhaustively enumerate all b or bi relations for those arcs. If e is the number of the arcs labeled by the $\{b, bi\}$ relation set, the total number of the enumeration will be 2^e.

For any STP network, we can associate it with a directed edge-weighted graph, G_d, called a *distance graph*. An STP is consistent if and only if the corresponding distance-graph G_d has no negative weight cycles. The *minimal network* of the STP can be specified by a complete directed graph, called the *d-graph*, where each edge, $i \rightarrow j$, is labeled by the shortest path length, d_{ij}, in G_d (Dechter et al. 1991). An STP network can be solved in $O(n^3)$ time by the Floyd-Warshall's all-pairs shortest-paths algorithm, where n is the number of variables in the STP network, Thus, the overall complexity of the brute-force method for the scheduling problem is $O(n^3 2^e)$.

We can increase the efficiency of the brute-force method by running a backtracking search on a *meta-CSP* network whose variables are the arcs in the GTCN which have the relation set $\{b, bi\}$, and whose domains are the two possible relations. In the backtracking procedure, we randomly select one variable and assign b or bi relation to that variable. If the corresponding STP network is consistent, we continue until we generate a solution; otherwise we backtrack.

More Effective Methods

As first observed by Erschler et al. (1976), the structure of resource capacity constraints - i.e. that $O_i\{b, bi\}O_j$ for any O_i and O_j competing for the same resource - can be exploited to define dominance conditions over the set of possible orderings in any feasible solution. In (Smith & Cheng 1993) a scheduling procedure called Precedence Constraint Posting (PCP) is defined which couples the use of such dominance checking with simple "slack-based" heuristics for variable and value ordering in the meta-CSP problem. The intuition behind these heuristics is simple. When faced with two or more unresolved ordering decisions, focus first on the decision with the least sequencing flexibility. Since any decision made is likely reduce the flexibility of those that remain, delaying the currently most constrained choice increases the chances of arriving at an infeasible state. In taking the decision, select the choice that retains the most sequencing flexibility (and thus leaves the search with the most degrees of freedom). On a set of previously published benchmark problems, PCP was shown to significantly outperform other contemporary approaches to the classical constraint satisfaction scheduling problem.

Unfortunately, temporal slack measures are based solely on the earliest start times and latest end times of operations, and do not reflect the influence of finite-interval separation constraints between operations. As

this type of constraint is added into the problem, we would thus expect slack measures to provide a less effective basis for estimating sequencing flexibility. However, our problem formulation within the GTCN model suggests a straightforward means of generalizing the PCP procedure. In the following subsections, we develop dominance conditions and search control heuristics for this extended class of scheduling problems.

Dominance Conditions

Suppose X_{ij} is a currently unassigned variable in the meta-CSP network representing the choice associated with the constraint $O_i \{b, bi\} O_j$, and consider the d-graph associated with the current partial solution. Let s_i, e_i, s_j, and e_j be the start and end points respectively of O_i and O_j, and further assume d_{ij} is the shortest path length from e_i to s_j and d_{ji} is the shortest path length from e_j to s_i. Four mutually exclusive cases can be identified:

Case 1. If $d_{ij} \geq 0$ and $d_{ji} < 0$, then $O_i \{b\} O_j$ must be selected.

Case 2. If $d_{ji} \geq 0$ and $d_{ij} < 0$, then $O_i \{bi\} O_j$ must be selected.

Case 3. If $d_{ji} < 0$ and $d_{ij} < 0$, then the STP network is inconsistent.

Case 4. If $d_{ji} \geq 0$ and $d_{ij} \geq 0$, then either relation is still possible.

Consider Case 1. Suppose that rather than selecting $O_i \{b\} O_j$, we select $O_i \{bi\} O_j$. This implies the insertion of a new edge $s_i \rightarrow e_j$ with 0 weight into the current distance graph G_d, corresponding to the linear inequality $e_j - s_i \leq 0$ (i.e., $s_i \geq e_j$). But, since G_d already contains a negative weight path from e_j to s_i (i.e., $d_{ji} < 0$), we now have a negative weight cycle in G_d. To avoid inconsistency, hence, we must select $O_i \{b\} O_j$. By similar arguments, we can derive Cases 2, 3, and 4.

These dominance conditions provide a direct basis for pruning with the meta-CSP search space. Detection of Case 1 or Case 2 in any state allows immediate assignment (and extension of the current STP network). Each time a new distance constraint is added to the STP network, the corresponding d-graph is determined. If Case 3 is detected in any state, the only recourse is to backtrack.

Variable and Value Ordering

The dominance conditions, of course, provide only necessary conditions for determining a set of feasible schedules. We are still left with the problem of resolving the undecided states specified by Case 4. In situations where application of the dominance conditions leaves the search in a state with several unassigned variables in the meta-CSP network, the shortest path information in the current d-graph provides estimates of the flexibility associated with each decision. Directly

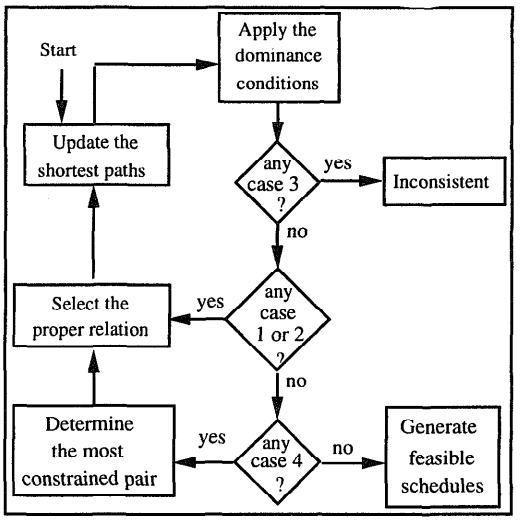

Figure 3: The SP-PCP Search Procedure

adapting the variable and value ordering heuristics defined by Smith and Cheng (1993), we define the *biased shortest path lengths* between O_i and O_j as

$$bd_{ij} = \frac{d_{ij}}{\sqrt{S}}; \quad bd_{ji} = \frac{d_{ji}}{\sqrt{S}}. \tag{1}$$

$$S = \frac{\min\{d_{ij}, d_{ji}\}}{\max\{d_{ij}, d_{ji}\}} \tag{2}$$

estimates the degree of similarity between the two values d_{ij} and d_{ji}. Given this definition of bd, the priority of a given decision variable X_{ij} is determined by the expression $pr_{X_{ij}} = -\min\{bd_{ij}, bd_{ji}\}$. The meta-CSP variable with the highest pr is selected as the next variable to assign.

Intuitively, this variable ordering heuristic is heavily oriented toward selection of the X_{ij} with the least sequencing flexibility, i.e., $min\{d_{ij}, d_{ji}\}$. The $\sqrt{S}$ bias is introduced to hedge in situations where the decision with the overall $min\{d_{ij}, d_{ji}\}$ has a very large $max\{d_{ij}, d_{ji}\}$, and a competing decision has two shortest path values just slightly larger than this overall minimum.

Having selected the next meta-CSP variable to assign, we commit to the ordering relation that retains the greatest flexibility. Specifically, if the highest priority variable is X_{ij}, then we select $O_i\{b\}O_j$ if $bd_{ij} > bd_{ji}$ and $O_i\{bi\}O_j$ otherwise.

Figure 3 graphically depicts the overall search procedure, which will be referred to as *SP-PCP*. We present SP-PCP as a backtrack-free approximation procedure. However, it should be clear how the procedure can be integrated into a backtracking search.

Performance Evaluation

In this section, we evaluate the performance of the shortest path dominance conditions and variable/value

ordering heuristics on a set of randomly generated problems. We contrast the performance of the SP-PCP with that of the original PCP developed in (Smith & Cheng 1993), to assess the performance gain directly attributable to reliance on shortest path information as opposed to slack information for search control. We also embed both SP-PCP and PCP within a chronological backtracking search. In this case, performance is contrasted with that of the basic GTCN backtracking search procedure previously given, to evaluate the performance leverage provided by both dominance checking and variable/value ordering in solving this class of problems.

We simulate a scheduling scenario where jobs require operations to be performed on each of 5 resources. We generate problem sets of 5 different sizes: 6 jobs (or 30 total operations), 8 jobs (40 operations), 10 jobs (50 operations), 16 jobs (80 operations) and 20 jobs (100 operations). For each problem size, we randomly generate 50 problem instances. This gives a total of 250 scheduling problems, with size ranging from very small to fairly large.

Operation sequences are randomly generated and each of the 5 resources must be visited once. Minimum processing times are drawn from a uniform distribution $U[10, 50]$, and the maximum processing times are generated by multiplying each minimum processing time by a random tolerance $(1 + t)$, with $t \sim U[0, 0.4]$. Job ready times are drawn from a uniform distribution $MU[0, 0.1]$, where M represents the minimum overall duration of the schedule (or "makespan").[1] Similarly, job deadlines are drawn from another uniform distribution $MU[1, 1.1]$.[2] Finally, we generate separation constraints between every two consecutive operations in each job. A separation constraint is represented by a random interval, $[a, b]$, with $a \sim U[0, 10]$ and $b \sim U[40, 50]$.

Six different procedures where applied to the generated set of 250 problems: the SP-PCP and original PCP approximation procedures, chronological backtracking search with random variable/value ordering (denoted below as CB), chronological backtracking augmented with SP dominance checking (denoted as CB w/ D), and backtracking search variants of both SP-PCP and PCP (denoted as CB w/ SP-PCP and CB w/ PCP respectively). All procedures were implemented in C and run on a SUN Sparc 10 workstation. For all backtracking algorithms, the maximum time allowed for solution of any problem was limited

[1] $M = (n-1)\overline{p_{bk}} + \sum_{i=1}^{m} \overline{p_i}$, where n is the number of jobs, m the number of resources, $\overline{p_{bk}}$ the average minimum processing time of operations on the bottleneck resource, and $\overline{p_i}$ the average minimum processing time of operations on resource i. The bottleneck resource bk is the resource with the maximum total amount of operation (minimum) processing time.

[2] This scheme for generating job ready times and deadlines is taken from (Ow 1985).

size	pcp	sp-pcp	cb	cb w/ d	cb w/ pcp	cb w/ sp-pcp
30	38	49	50	50	50	50
40	16	46	32	50	50	50
50	0	40	10	28	46	50
80	0	37	0	1	2	48
100	0	33	0	0	0	46

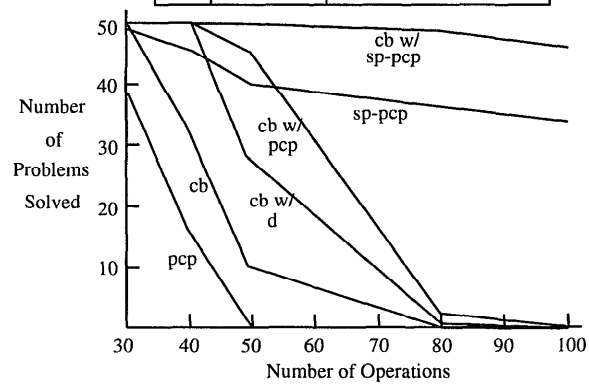

Figure 4: Number of problems solved

to 1,000 CPU seconds. Backtracking search algorithms were found to be capable of searching approximately 300,000 states in this time frame. Performance was measured in terms of number of problems solved and average solution time.

Figure 4 shows the number of problem instances solved in each problem size category by each procedure tested. The average time required (in CPU seconds) for solving all instances of each size category is given in Figure 5 (graphed on a logarithmic scale). For the backtracking procedures, this average time thus gives a lower bound on solution time for those categories in which all problems are not solved.

It is clear from the results obtained with SP-PCP and PCP (the backtrack-free procedures) that shortest path information provides a much more effective basis for directing the search. PCP was reasonably effective only on the smallest problem set and was not able to solve any problem with 50 or more operations. SP-PCP, alternatively, performed extremely well on the smaller problem sets. Though its effectiveness also diminishes with increasing problem size, it is, nonetheless, able to solve 33 of the 50 100-operation problems, with an average solution time of under 10 seconds.

SP-PCP results are more striking when considered in relation to the results produced by the backtracking procedures tested. We see that at small problem sizes (30-40 operations), all algorithms perform very well. As problem size increases, the exponentially increasing size of the search space quickly overwhelms the basic chronological backtracking procedure CB and its performance deteriorates drastically. The incorporation of dominance condition checking (CB w/ D) is seen to improve the performance of CB on most problem sets. However, it is interesting to note that incorpo-

size	pcp	sp-pcp	cb	cb w/ d	cb w/ pcp	cb w/ sp-pcp
30	0.2	0.2	8.4	1.4	0.4	0.4
40	0.4	0.5	518.8	75.8	2.6	1.1
50	0.8	1.0	860.9	506.0	178.9	5.0
80	2.1	4.4	1005	983.5	978.4	88.9
100	3.3	9.2	1006	1006	1005	154.1

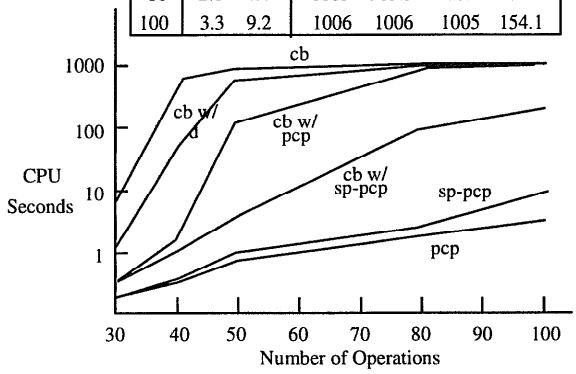

Figure 5: CPU seconds used

ration of the original PCP procedure yields a somewhat larger performance improvement. This indicates that the look-ahead advantage provided by slack-based variable/value ordering in fact outweighs the performance gains offered by stronger search space pruning conditions. In the case of both of these augmented backtracking procedures, however, the larger, 80-100 operation problems generally cannot be solved.

The results obtained by integrating SP-PCP into the chronological backtracking procedure further indicate the impact of our simple variable and value ordering heuristics on problem solving performance. Not only is this procedure effective across problem sets of all sizes, solving all but 4 problem instances, but it is also very efficient. The use of shortest path information quickly moves the search into profitable regions of the space, and dramatically reduces the need for backtracking.

Conclusions

In this paper, we have presented an efficient procedure for solving scheduling problems that are complicated by finite-interval separation constraints on the execution of different operations. Despite the practical importance of this class of problems in many application domains, it has received little attention in the scheduling research community. Following recent research in constraint-posting scheduling, we formulated the scheduling problem as one of establishing before or after relations between pairs of operations that require the same resource. The problem was characterized as a general temporal constraint network (GTCN), and, using properties of Simple Temporal Problems, we were able to directly generalize a high performance scheduling procedure previously developed for a more restricted class of scheduling problems. We established dominance conditions, based on shortest path information, for early pruning of infeasible regions of the meta-CSP search space. We also developed heuristics for variable and value ordering in the meta-CSP search, based on use of shortest path information as an estimation of decision flexibility. Experimental evaluation of these heuristics on a set of randomly generated problems drawn from a manufacturing scenario indicated significant performance improvement in relation to general GTCN solution procedures and the original constraint-posting procedure.

References

Allen, J. F. 1983. Maintaining knowledge about temporal intervals. *CACM*, 11(26), 832-843.

Dean, T. and McDermott, D. 1987. Temporal data base management. *Artificial Intelligence*, 32, 1-55.

Dechter, R., Meiri, I., and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence*, 49, 61-95.

Erschler, J., Roubellat, F., and Vernhes, J. P. 1976. Finding some essential characteristics of the feasible solutions for a scheduling problem. *Operations Research*, 24, 772-782.

Garey, M. R. and Johnson, D. S. 1979. *Computers and Intractability, a Guide to the Theory of NP-Completeness*, W.H. Freeman Company.

Kautz, H. and Ladkin, P. B. 1991. Integrating metric and qualitative temporal reasoning. In *Proc. of AAAI-91*, 241-246. Anaheim, CA.

Keng, N. and Yun, D. Y. Y. 1989. A planning/scheduling methodology for the constrained resource problem. In *Proc. of IJCAI-89*, Detroit, MI.

Ladkin, P. B. and Maddux, R. D. 1989. On binary constraint networks. Technical report, Kestrel Institute, Palo Alto, CA.

Meiri, I. 1991. Combining qualitative and quantitative constraints in temporal reasoning. In *Proc. of AAAI-91*, 260-267. Anaheim, CA.

Minton, S., Johnston, M., Philips, A. B., and Laird, P. 1992. Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence* 58, 161-205.

Muscettola, N. 1993. Scheduling by iterative partition of bottleneck conflicts. In *Proc. of 9th IEEE Conf. on AI Applications*, Orlando, FL.

Ow, P. S. 1985. Focused Scheduling in Proportionate Flowshops. *Management Science* 31 (7) 852-869.

Sadeh, N., and Fox M., 1990. Variable and value ordering heuristics for activity-based job-shop scheduling. *4th Int. Conf. on Expert Systems in Prod. and Oper. Management*, 134-144, Hilton Head Is., SC.

Smith, S. F. and Cheng, C. 1993. Slack-based heuristic for constraint satisfaction scheduling. In *Proc. of AAAI-93*, 139-144. Washington, DC.

Vilain, M. and Kautz, H. 1986. Constraint propagation algorithms for temporal reasoning. In *Proc. of AAAI-86*, 377-382, Philadelphia, PA.