A Hybrid Approach to Scheduling with Earliness and Tardiness Costs

J. Christopher Beck[1] and Philippe Refalo[2]

[1] ILOG SA
9, rue de Verdun, BP85,
94253 Gentilly Cedex, France
cbeck@ilog.fr
[2] ILOG, SA
Les Taissounières
1681, route des Dolines
Sophia-Antipolis 06560 VALBONNE, France
refalo@ilog.fr

Abstract. A hybrid technique using constraint programming and linear programming is applied to the problem of scheduling with earliness and tardiness costs. The linear model maintains a set of relaxed optimal start times which are used to guide the constraint programming search heuristic. In addition, the constraint programming problem model employs the strong constraint propagation techniques responsible for many of the advances in constraint programming for scheduling in the past few years. Empirical results validate our approach and show, in particular, that creating and solving a subproblem containing only the activities with direct impact on the cost function and then using this solution in the main search, significantly increases the number of problems that can be solved to optimality while significantly decreasing the search time.

1 Introduction

It has long been recognized that constraint programming (CP) techniques and linear programming (LP) techniques have complementary strengths and weaknesses. Optimization techniques based on linear relaxations perform well in reasoning about costs, providing an optimal solution of a linear relaxation of the problem. Sophisticated search strategies based on this solution can be used to guide the search toward optimal solutions for the full problem. Dynamic problem reformulation such as the generation of cutting planes can also be employed to tighten the linear formulation, resulting in a relaxed optimal solution which is closer to a solution of the full problem.

Constraint programming techniques make inferences about relationships among problem objects using the structural information of the problem formulation. While these inferences often take the form of domain reductions, they can more generally be seen as adding implied constraints to the evolving problem representation [2]. In constraint based scheduling, for example, it is common to infer precedence constraints which state that one activity must start after the completion of another activity. The addition of constraints to the problem model can also be used as a branching rule [22, 4].

The gain in merging two techniques with complementary strengths is that each solver can derive information that is not easily available to the other. By sharing this information between the solvers, stronger problem solving performance can be achieved. Cooperative solvers have been shown to be effective in solving a number of combinatorial optimization problems (see [5, 7, 14, 17–19]).

This paper describes a hybrid approach to solving scheduling problems with earliness and tardiness costs. Unlike classical scheduling problems (e.g., job shop [6]), the optimization function is to minimize the weighted difference between the due date of a job and its actual completion time. Depending on whether the job finishes before the due date or after the due date, different cost functions apply. Given the importance of "just-in-time" inventory management, problems of this type are frequently encountered in industrial settings.

The problem is solved with a branch and bound approach that uses a linear programming solver and a constraint-based scheduler at each search node. The latter infers reductions in the possible start times of activities and precedence constraints between activities while the former maintains an optimal solution to a linear relaxation of the problem. The relaxed optimal solution does not necessarily satisfy the resource requirements, however it provides a basis for the search heuristic. Using the relaxed optimal solution, we identify sets of activities for which the relaxed optimal start times lead to the breakage of a resource capacity constraint. We then branch by posting a precedence constraint between a pair of conflicting activities. This precedence constraint is added to both the constraint-based scheduler and the linear programming model so that constraint propagation can be performed and an updated relaxed optimal solution can be found.

From a operations research perspective, there is a similarity between this branching rule and techniques for mixed integer programming (MIP). MIP techniques tend to focus on violated integrality conditions, guiding heuristic search with the characteristics of these violations. In our approach, we use the violated resource capacity constraints to guide heuristic search. From a scheduling perspective, the use of texture measurements [4, 1, 3, 20] and resource profiles [8–10] to estimate resource usage and guide heuristic search is a well-established technique.

The primary contribution of this paper and the technique that results in significant gains in problem solving performance is the identification of a *cost relevant subproblem* (CRS) that can be solved at each node in the search. The solution to this subproblem is then used either to quickly find a solution to the overall problem (if the subproblem solution can be extended) or to provide an updated lower bound on the optimization function.

Our hybrid approach builds on *probe backtrack search* [12] which was originally applied to solving a rescheduling problem with minimal perturbation (see Section 10). In particular, we adopt the probe backtrack method of interaction between the CP solver and the LP solver. Differences in this paper include the use of the hybrid technique for the solving of scheduling problem from scratch (rather than rescheduling) and the use of the cost relevant subproblem.

2 Search Overview

Before presenting the details of the problem and problem solving approach, in this section, we provide an overview of the two search techniques used in this paper.

The first technique, denoted *Probe*, is essentially an adaptation of probe backtrack search [12] to the problem of scheduling with earliness and tardiness costs. Search is conducted as follows:

1. At a search state, constraint propagation, including propagation of global scheduling constraints, is performed.
2. An LP representation of the search state (including the information generated by constraint propagation) is solved, generating relaxed optimal start times for the activities.
3. For each resource in the scheduling problem, resource profiles are formed assuming that each activity starts at its relaxed optimal start time as generated by the LP solution. These profiles represent the actual resource usage (ignoring resource capacity constraints) if the relaxed optimal start times were to be assigned to each activity.
4. A heuristic decision (i.e., a branch) is made by identifying a time point at which the relaxed optimal start times over-capacitate a resource. The heuristic decision takes the form of adding a precedence constraint between a pair of activities which compete for the resource at the identified time point. The reverse precedence constraint is used on backtracking.[1]
5. Goto 1.

The second technique, denoted *Probe+CRS* uses the *Probe* technique as the main search, but adds the solving of a *cost relevant subproblem* (CRS) at each search state. When search enters a new search state, before the *Probe* steps enumerated above, the *Probe+CRS* technique performs the following steps:

1. A CRS is created and solved. The CRS contains the set of resource and all activities which have a direct impact on the optimization function. The subproblem is solved to optimality (using the *Probe* technique) and the start times of each activity and the cost is saved.
2. Returning to the full problem, we check if the subproblem solution can be extended. This is done by assigning the start times generated by the subproblem solution and performing a pure CP scheduling search to see if these start times can lead to a global solution. This CP search is still NP-hard, however, the start times from the subproblem are very constraining and so it is likely (and, indeed, required, for reasonable performance of this approach) that a global solution consistent with the start times will be found or shown not to exist with very few branches.
3. If a global solution does exist, we have found an optimal solution for the subtree rooted at the current node and so we can backtrack, continuing the main branch-and-bound *Probe* search.

[1] As we limit the problems addressed in this paper to have only unary resources (i.e., resources that can perform only one activity at a time), this branching scheme is clearly complete.

4. If a global solution does not exist, the cost of the subproblem represents a lower bound on the cost of solutions in the subtree rooted at the current node. Specifically all solutions must have a cost greater than or equal to the sum of the subproblem costs.
5. Goto the first step of the *Probe* procedure.

3 Problem Definition

An $n \times m$ earliness/tardiness scheduling problem (ETSP) consists of a set, J, of n jobs and a set, R of k resources. For each job j there is an ordered set of activities $A(j) = \{a_{j1}, a_{j2}, \ldots, a_{jm}\}$. Each activity is characterized by its start time $s(a_{ji})$ and its processing time $pt(a_{ji})$. Let $s_{min}(a_{ji})$, represent the minimum start time of activity a_{ji} and $s_{max}(a_{ji})$ represent its maximum start time.

No pre-emption is allowed, meaning that once an activity has started execution, it must execute for its entire duration without interruption.

For each job, $j \in J$, there is a due date, dd_j, and two cost factors, earliness, ec_j, and tardiness, tc_j. Earliness is the cost factor used if the last activity a_{jm} of job j ends before its due date while tardiness is the cost factor used if it ends after its due date. More precisely the cost incurred for each job j is

$$c_j = \begin{cases} ec_j(dd_j - l_j) & \text{if } l_j \leq dd_j \\ tc_j(l_j - dd_j) & \text{if } l_j > dd_j \end{cases} \tag{1}$$

Where l_j is the completion date of last activity a_{jm}. That is, $l_j = s(a_{jm}) + pt(a_{jm})$. Observe that this cost is 0 if the job finishes on time, that is, if $l_j = dd_j$.

The cost for the entire problem, c_J, is the sum of the costs incurred for each job, $j \in J$:

$$c_J = \sum_{j \in J} c_j \tag{2}$$

In the most general version of the ETSP, for each activity, a_{ji}, and resource, $r_h \in R$, let $rcap(a_{ji}, r_h)$ represent the capacity of r_h which is required by a_{ji}. This value is given as input and varies between 0 and the capacity of the resource, which is defined as follows. For each resource, $r_h \in R$, there is a predefined capacity, $cap(r_h)$, which is the maximum amount of the resource that is available at any time point.

Finally, we also have a set of resource constraints which express that the capacity required by a set of activities executing on a resource at each time point must be equal to or less than the total resource capacity.

A solution to an ETSP problem is to assign start times to each activity such that:

1. the activities in a job are processed in the specified order
2. for each resource, the resource capacity constraints are respected
3. the total cost of the problem, c_J, is minimized.

For this paper, we place the following further restrictions on the ETSP model:

– There are m resources each of which has a capacity of one.

- Each activity has a non-zero required capacity for only one resource. For that one non-zero resource the required capacity is one.
- Each of the m activities in a job requires a different resource.

These restrictions mean that the ETSPs addressed in this paper are extensions of the job shop scheduling problem which is known to be NP-hard [13].

4 A Linear Relaxation

To formulate a linear relaxation of the restricted ETSP problem, for each activity a_{ji} we introduce a variable x_{ji} representing the start time of the activity. We also introduce a variable y_j, $j \in J$ that represents the cost associated with job j. The objective function is thus

$$\text{minimize} \sum_{j \in J} y_j$$

and the constraints that define the problems are:

- All activities must start between their minimum and maximum start time

$$s_{min}(a_{ji}) \leq x_{ji} \leq s_{max}(a_{ji}) \text{ for } j \in J \text{ and } i \in \{1, \ldots, m\}$$

- Each activity of a job must start after the completion of the previous one

$$x_{ji} + pt(a_{ji}) \leq x_{ji+1} \text{ for } j \in J \text{ and } i \in \{1, \ldots, m-1\}$$

- Each variable y_i is equal to the cost incurred if the corresponding job does not finish at its due date

$$y_i \geq tc_j(x_{jm} + pt(a_{jm})) - tc_j dd_j$$
$$y_i \geq ec_j dd_j - ec_j(x_{jm} + pt(a_{jm}))$$

  This relaxation is simply the one obtained by linearizing the piecewise linear expression (1) as described more generally in [17].

The linear relaxation we use does not take into account the resource capacity constraints. Since we have a convex cost function and since the problem matrix is integer and totally unimodular, using a simplex method for solving this model gives integer solutions. Consequently, this solution may violate some resource capacity constraints. The heuristic described below is responsible for resolving conflicts between activities.

5 Problem Solving

The problem described above is solved using a branch and bound technique. At each node of the search tree, both a linear programming solver (ILOG Hybrid 1.1/ILOG Cplex 7.1 [15, 11]) and a constraint based scheduling solver (ILOG Scheduler 5.1 [21]) are used.

The constraint based scheduler uses the edge finder [16] and a precedence graph constraint [21] to infer new bounds on variables and precedence constraints between activities. These constraints are transmitted to the linear programming solver.

The relaxed optimal solution given by the linear programming solver gives an integer solution for the start time of each activity as the LP problem is totally unimodular. This solution does not necessarily satisfy the resource capacity constraints but rather is used to determine conflicting activities with which a branch is formed using the strategy described below.

5.1 The Scheduling Heuristic

The heuristic investigated in this paper is based on the creation of resource profiles, the identification of peaks in these profiles which break the resource capacity, and the sequencing of a pair of activities that contribute to the peak.

Resource profiles are curves, over time, that estimate the capacity required of a resource. Different assumptions made when building the curves result in different estimations (e.g., probabilistic estimates [4]).

In this paper, we make the assumption that each activity will start at a time defined by the solution to the LP relaxation of the scheduling problem. We generate a resource profile by placing each activity at its start time ignoring resource capacity constraints. For example, a set of activities has start times (in the LP) and durations as displayed in Figure 1(a). The corresponding resource profile is displayed in Figure 1(b). If the displayed resource is a unary resource, assigning the current start times will not lead to a feasible solution.

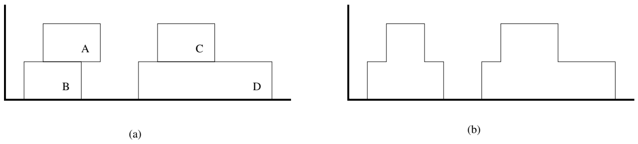

Fig. 1. Given the activities in (a), with their indicated start times and durations, the resource profile for the resource is shown in (b).

The LP solution found at search node, S, provides an optimal start time for each activity when resource capacity constraints are not present. If these relaxed optimal start times satisfy the resource capacity constraints, we have found an optimal solution in for the search subtree rooted at S.

Because the relaxed optimal start times do not necessarily lead to a solution that respects the resource capacity constraints, we branch by adding precedence constraints that move toward a feasible solution. Given, the relaxed optimal start times, our heuristic is straightforward:

– build the resource profiles and identify a peak (a time point where the resource capacity constraint is broken),

– identify the set of activities contributing to the peak,
– post a precedence constraint between one pair of the competing activities in order to reduce the resource contention.

For now, we have put little effort into peak selection: we randomly choose a resource, R, and time point, tp, such that the resource profile is greater than the resource capacity on R at tp.

We then select an activity pair by first looking at the cost status of each activity contributing to the resource peak. In an ETSP, we can partition the activities into two sets: cost relevant and non-cost relevant activities. The former set contains all the activities with a direct impact on the cost function. With only earliness and tardiness costs, the last activity in each job has such an impact. The latter set contains all the other activities.

We identify all contributing activities which are cost relevant. If there are two or more such activities, we choose the pair with the highest potential cost impact, found by summing the earliness and tardiness factors of that activity. A minimum (local) cost precedence constraint is then chosen. Given the predefined earliness and tardiness cost factors it is straightforward to choose the sequence with the lowest local cost impact. The reverse precedence constraint is used on backtracking.

If one or fewer of the activities contributing to a resource peak is cost relevant, we turn to the slack-based heuristic due to Smith & Cheng [22]. For all pairs of activities that have not been sequenced, we calculate the slack of each possible precedence constraint and post the constraint which preserves the greatest amount of slack. Again the reverse precedence constraint is used on backtracking.

An important part of this heuristic is that the precedence constraint is added to both the CP and LP models. This enables the CP model to deduce new implied constraints through the propagation of the precedence constraint and it ensures that the optimal start times of the cost relevant activities will be appropriately updated when solving the LP in the child nodes.

6 A Cost Relevant Subproblem for ETSPs

The fact that only a subset of the activities are cost relevant in an ETSP allows the simple definition of a cost relevant subproblem. The *cost relevant subproblem* (CRS) of an ETSP contains the resources and cost function of the full ETSP but only the cost relevant activities. A CRS is itself an instance of an ETSP with each job containing one activity. Therefore, the *Probe* technique can be applied without modification to a CRS. A solution to a CRS provides two pieces of information that can be used in solving the full problem. First, it provides a lower bound on the cost of the full problem and, second, it provides a set of assignments for the cost relevant activities. If this set of assignments leads to a feasible solution to the full problem, we have found an optimal solution at the current search node.

Given a search state, S, we generate a CRS using the resources, cost function, and cost relevant activities of the full problem. The time windows of the activities in the CRS are the corresponding time windows of those activities in the full problem at state S. Two scheduling problems are then solved:

1. The CRS is solved, using *Probe*.
2. The full model is solved using the start times for the cost relevant activities from the solution to the CRS. Because we already have assignments for the cost relevant activities, we search for a feasible solution using a simple, pure CP search strategy which ranks the activities on each resource.

If the second problem is successfully solved, we have found an optimal solution in the search subtree rooted at S. We can, therefore, continue the branch and bound procedure, by returning to the parent of S. In contrast, if no solution exists for the second problem, we can use the cost of the solution of the CRS as a lower bound on the optimal solution in the search subtree rooted at S: all solutions in the subtree below S must have a cost greater than or equal to the cost of the subproblem solution. However, as we have no solution to the full model, we must continue exploration of the subtree below S.

It should be pointed out that the lower bound cannot be added as a constraint to the model because it may destroy the unimodularity of the LP model. To take a trivial example, assume that one activity, A, has a due date of 10 and a tardiness cost with a slope of 2. If we constrain the lower bound on the cost function to be 1, we now have a non-integer optimal start time for A: 10.5. Therefore, rather than adding the lower bound to the model, we simply compare it to any solutions that are found. If we find a solution with a cost equal to the lower bound, we can immediately return to the parent node, continuing the branch-and-bound as we have found an optimal solution for the subtree at the current search node.

If no solution exists to the CRS, we can also conclude that no solution exists in the subtree below S. The CRS is a subproblem with a subset of the constraints in the global problem. Therefore, if no solution exists for the subset of constraints in the CRS at node S, no solution exists in the global problem at node S.

The usefulness of this technique depends strongly on the effort required to solve the CRS and the subsequent effort required to solve the full problem using the start time assignments from the CRS solution. We can expect the effort required to solve the CRS to be much less than that of solving the full problem because of the much smaller number of activities in the CRS. We also expect that solving the full problem using the start time assignments from the CRS will be fast as we are dealing with a very constrained problem and so the global scheduling constraint propagation will quickly guide the search to a solution or to a proof that one does not exist. As demonstrated below, both of these expectations are supported by our experimental results.

7 Experiments

The goals of our experiments are to validate the use of the *Probe* technique for scheduling with earliness and tardiness costs and to gather initial results regarding the usefulness of the cost relevant subproblem.

The CRS can be applied in any search state. In these experiments, however, we restrict ourselves to using it only in the initial search state of the full problem. This means that if the CRS at the root node cannot be extended to a full solution, the rest of the search is conducted using the *Probe* technique plus the lower bound provided by the CRS solution at the root node.

Future work will investigate the use of the CRS at multiple nodes in the search.

7.1 Experimental Problems

We generated nine sets of problems each with 10 problem instances as follows. Using the job shop problem generator of Watson et al. [25], we generated 10 job shop problems in each of three sizes: 10×10, 15×10, and 20×10. Each problem instance has a workflow structure and has the duration of each activity drawn with uniform probability from the interval [1, 99]. The workflow structure means that the resources in the problem is partitioned into two sets of approximately equal size. The activities in each job, then, must use each of the resources in the first partition before using any of the resources in the second. Such job shop problems are believed to be more difficult than problems without the workflow structure.

For each job shop problem instance, three earliness/tardiness problems are generated with different due dates and costs. The costs are uniformly drawn from the interval [1, 20]. The due dates are set by calculating, tlb, a lower bound on the job shop problem due to Taillard [24]. The due date for each job was then uniformly drawn from the interval:

$$[0.75 \times tlb \times lf, 1.25 \times tlb \times lf] \tag{3}$$

Where lf is the *looseness factor* and with possible values of $\{1.0, 1.3, 1.5\}$.

For each problem size and looseness factor, 10 earliness/tardiness cost problems are generated. This results in nine sets of problems each with 10 problem instances.

7.2 Hardware and Software Details

All experiments were run on an UltraSparc10, running SunOS 5.7, with 256M of RAM. For each problem instance, we imposed a maximum time limit of 1200 seconds.

All algorithms are implemented using ILOG Scheduler 5.1, ILOG Hybrid 1.1, and ILOG CPLEX 7.1.

8 Results

In this section, we present two views of the same results. In Table 1 we present the number of problems solved to optimality and the mean CPU time for each looseness factor. The two solution techniques used are probe backtracking (*Probe*) and probe backtracking plus the cost relevant subproblem at the root node (*Probe+CRS*). Each table cell represents the results of runs on 30 problems, 10 of each size. It should be noted that the mean CPU time is somewhat misleading as it is dominated by those problems for which the optimal solution could not be found. With the exception of looseness level 1.0, *Probe+CRS* is able to solve and prove the optimality of all problems. The maximum CPU time for these problems was 1.26 seconds. In contrast, only a small number of the problems can be solved without the CRS.

Looseness	1.0		1.3		1.5	
	# Solved	Mean CPU	# Solved	Mean CPU	# Solved	Mean CPU
Probe	0	1200	2	1127.03	10	827.92
Probe+CRS	7	920.12	30	0.31	30	0.28

Table 1. The number of problems solved to optimality (out of 30) and the mean CPU time for probe backtracking (*Probe*) and for probe backtracking plus the cost relevant subproblem (*Probe+CRS*) for different levels of the looseness factor.

Table 2 presents the same data as Table 1 from the perspective of problem size. Again each cell represents the results of 30 problems, 10 problems at each looseness factor. Note that with the CRS technique many of the problems at each size can be solved and further that performance of the search using CRS increases on larger problems.

Size	10×10		15×10		20×10	
	# Solved	Mean CPU	# Solved	Mean CPU	# Solved	Mean CPU
Probe	6	967.81	3	1101.84	3	1085.30
Probe+CRS	20	400.10	24	240.22	23	280.38

Table 2. The number of problems solved to optimality (out of 30) and the mean CPU time for probe backtracking (*Probe*) and for probe backtracking plus the cost relevant subproblem (*Probe+CRS*) for problems of different size.

9 Discussion

9.1 The Cost Relevant Subproblem

It is clear from the results that the CRS technique significantly adds to the number of problems that can be solved and the speed at which they can be solved.

There are two trends in the results that deserve explanation. First, in Table 1 it can be seen that less than 30% of the problems at looseness level 1.0 are solved while all problems at the other looseness levels are solved. This disparity results from the fact that with a small looseness factor, it is unlikely that the start times found in solving the CRS at the root node can be extended to a full solution. The start times assigned are such that they can not be met. In contrast, with higher levels of looseness the start times from the CRS solution can be easily extended to a overall solution. This suggests that solving the cost relevant subproblem *only* at the root node is not that useful for tight problems. Even so, seven problems at looseness level 1.0 were solved, while without solving the CRS, no problems were solved.

The second trend can be seen in Table 2. Here the *Probe+CRS* is able to solve more problems among the larger problems sets than on the 10×10 problems. This initially appears strange especially as without CRS, more of the small problems are solved than the large problems. Recall that the Taillard lower bound was used as the center of the interval from which the due dates were uniformly drawn. It has been previously noted [24] that for an $n \times m$ job shop problem, for constant m, as n is increased the probability that the Taillard lower bound is equal to the optimal makespan approaches 1. This means that as the problem size grows, the actual looseness of a problem at level 1.0 increases. We saw in Table 1 the positive effect of looseness on the success of the CRS technique.

Another observation about the search behavior using the CRS technique is that either the CRS solution was easily extended to a global solution or no globally optimal solution was found in the subsequent search using *Probe*. That is, if the CRS solution could not be directly extended, the *Probe* search, limited to 1200 seconds, could not find an optimal solution for any problems.

Our experimental results show that both problem solving ability and speed increase substantially when the CRS solution is used at the root node. While loose problems are solved easily using CRS, it is perhaps the tighter problems that are more industrially interesting. Therefore, part of our future work will be to investigate the use of the CRS technique at search nodes other than the root node.

9.2 The Linear Model

One of the goals of our experimentation was to validate the hybrid scheduling model of probe backtracking in a new problem domain. The results clearly show that these hybrid scheduling techniques can lead to strong scheduling performance.

The question arises, however, as to the usefulness of using a full LP solver. In this paper, the only role of the linear model was to supply the relaxed optimal start times (at each node) of the cost relevant activities. This is critical for the correctness of the technique presented here but, given the form of the earliness and tardiness costs, there exists an efficient algorithm that can fill this role without requiring a full linear solver [23].

There are two main reasons we have chosen to use a full linear solver.

1. Generality: Clearly, the earliness/tardiness cost function investigated in this paper is not the only interesting cost function. With a full linear solver, we can easily add new cost components (e.g., transition cost between activities) provided that they can be expressed as part of a totally unimodular linear program. In contrast, it is unclear if the other techniques for maintaining the relax optimal start times can be easily extended to new cost criteria.
2. Problem Solving Power: There are a variety of existing techniques that can be explored to improve the linear model of the scheduling problem. As noted, there is currently no representation of the resource constraints in the linear model. However, it may be the case that through the introduction of a tighter linear relaxation, generation of cutting planes, and the use of other LP search methods, the linear model can contribute more than simply optimal start times. We intend to pursue the development of a tighter linear relaxation and investigate its impact on the scheduling performance.

9.3 The Scheduling Heuristic

Finally, it should be noted that no experimentation was done with the heuristic used to select resource peaks and to post precedence constraints. In particular, the peak selection technique and the selection of a precedence constraint on which to branch may benefit from the ideas presented in Cesta et al. [9].

We have made no assumptions in the heuristic that limits its application to unary capacity resources. The building of the resource profiles, the selection of a resource peak, and the sequencing of a pair of activities can be directly applied to resources with capacities greater than one.

10 Related Work

As noted the hybrid search technique presented here builds on the work of El Sakkout & Wallace [12] who present a CP/LP hybrid approach, called *probe backtrack search*, for rescheduling with the least disruption. El Sakkout & Wallace, formulate a hybrid CP/LP model where the CP model contains the original scheduling problem plus the new resource capacities resulting from a resource breakdown and the LP model contains a linear representation of each activity and a cost function. The cost function used is the sum, over all activities, of the absolute difference between the start time of an activity in the initial schedule and its start time in the repaired schedule. Search consists of using the relaxed optimal start times provided by the LP solution to identify time points where the new resource constraints are broken. The set of activities contributing to a break are then analyzed and a branch is formed by posting a precedence constraint between a pair of the competing activities.

This approach is embedded in a branch-and-bound algorithm to search for the optimal solution. A critical component of this approach is that the LP model is totally unimodular, meaning that the relaxed optimal start times generated by solving the LP are always integral.

11 Conclusion

In this paper, we presented a hybrid scheduling technique combining constraint programming and linear programming to solve earliness/tardiness scheduling problems. While the CP model represents the entire scheduling problem, the LP model only represents the activities and temporal network without the resource capacity constraints. At each node in the search the LP model provides the relaxed optimal start times for the activities and these start times are used as a basis for the CP scheduling heuristic.

Experimental results indicated that the use of a cost relevant subproblem of the earliness/tardiness scheduling problem significantly increases the number of problems which could be solved to optimality as well as the speed at which the problems could be solved.

We view this work as preliminary. There appears to be significant opportunities to improve scheduling performance through:

- tightening of the relaxation of the scheduling problem represented in the linear model.
- improvements in the CP-based heuristic.
- solving the cost relevant subproblem at multiple nodes in the search tree.

Nonetheless, this work demonstrates that scheduling problems including costs can be successfully solved with a CP/LP hybrid search technique.

References

1. J. C. Beck. *Texture measurements as a basis for heuristic commitment techniques in constraint-directed scheduling*. PhD thesis, University of Toronto, 1999.
2. J. C. Beck, A. J. Davenport, E. D. Davis, and M. S. Fox. The ODO project: Toward a unified basis for constraint-directed scheduling. *Journal of Scheduling*, 1(2):89–125, 1998.
3. J. C. Beck and M. S. Fox. Constraint directed techniques for scheduling with alternative activities. *Artificial Intelligence*, 121:211–250, 2000.
4. J. C. Beck and M. S. Fox. Dynamic problem structure analysis as a basis for constraint-directed scheduling heuristics. *Artificial Intelligence*, 117(1):31–81, 2000.
5. H. Beringer and B. de Backer. Combinatorial problem solving in constraint logic programming with cooperating solvers. In *Logic Programming : Formal Methods and Practical Applications*. Elsevier Science Publishers, 1994.
6. J. Blazewicz, W. Domschke, and E. Pesch. The job shop scheduling problem: Conventional and new solution techniques. *European Journal of Operational Research*, 93(1):1–33, 1996.
7. A. Bockmayer and T. Kasper. Branch and infer: A unifying framework for integer and finite domain constraint programming. *INFORMS Journal on Computing*, 10(3):287–300, 1998.
8. A. Cesta, A. Oddi, and S. Smith. An Iterative Sampling Procedure for Resource Constrained Project Scheduling with Time Windows. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, 1999.
9. A. Cesta, A. Oddi, and S. Smith. A constraint-based method for project scheduling with time windows. *Journal of Heuristics*, 2000. to appear.
10. A. Cesta, A. Oddi, and S. F. Smith. Profile-based algorithms to solve multiple capacitated metric scheduling problems. In R. Simmons, M. Veloso, and S. F. Smith, editors, *Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems*, pages 214–223, Menlo Park, CA, 1998. AAAI Press.
11. CPLEX. *ILOG CPLEX 7.0 User's Manual*. ILOG, S.A., 1999.
12. H. El Sakkout and M. Wallace. Probe backtrack search for minimal perturbation in dynamic scheduling. *CONSTRAINTS*, 5(4):359–388, 2000.
13. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.
14. J. N. Hooker, G. Ottosson, E. S. Thornsteinsson, and H.-J. Kim. A scheme for unifying optimization and constraint satisfaction methods. *Knowledge Engineering Review*, 2000. to appear.
15. Hybrid. *ILOG Hybrid 1.0 User's Manual*. ILOG, S.A., 2000.
16. W. P. M. Nuijten. *Time and resource constrained scheduling: a constraint satisfaction approach*. PhD thesis, Department of Mathematics and Computing Science, Eindhoven University of Technology, 1994.
17. P. Refalo. Tight cooperation and its application in piecewise linear optimization. In *Proceedings of Fifth International Conference on Principles and Practice of Constraint Programming (CP'99)*, pages 369–383, Alexandria, Virginia, October 1999. Springer-Verlag.

18. P. Refalo. Linear formulation of constraint programming models and hybrid solvers. In *Proceedings of Sixth International Conference on Principles and Practice of Constraint Programming (CP 2000)*, pages 369–383, Singapore, September 2000. Springer-Verlag.
19. R. Rodosek and M. Wallace. A generic model and hybrid algorithm for hoist scheduling problems. In *Proceedings of the Fourth International Conference on Principles and Practice of Constraint Programming (CP'98)*, pages 385 – 399, Pisa, Italy, 1998. Also in LNCS 1520.
20. N. Sadeh and M. S. Fox. Variable and value ordering heuristics for the job shop scheduling constraint satisfaction problem. *Artificial Intelligence*, 86(1), 1996.
21. Scheduler. *ILOG Scheduler 5.1 Users' Manual and Reference Manual*. ILOG, S.A., 2001.
22. S. F. Smith and C. C. Cheng. Slack-based heuristics for constraint satisfaction scheduling. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pages 139–144, 1993.
23. F. Sourd. *Study about solving disjunctive Scheduling Problems*. PhD thesis, Université Paris VI, 2000.
24. E. Taillard. Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64:278–285, 1993.
25. J. Watson, L. Barbulescu, A. Howe, and L. Whitley. Algorithms performance and problem structure for flow-shop scheduling. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, pages 688–695, 1999.