# Constraint Programming Search Procedure for Earliness/Tardiness Job Shop Scheduling Problem

**Jan Kelbel** and **Zdeněk Hanzálek**
Centre for Applied Cybernetics, Department of Control Engineering
Czech Technical University in Prague, Czech Republic
{kelbelj,hanzalek}@fel.cvut.cz

**Abstract**

This paper describes a constraint programming approach to solving a scheduling problem with earliness and tardiness cost using a problem specific search procedure. The presented algorithm is tested on a set of randomly generated instances of the job shop scheduling problem with earliness and tardiness costs. The experiments are executed also for three other algorithms, and the results are then compared.

## Introduction

Scheduling problems with storage costs for early finished jobs and delay penalties for late jobs are common in industry. This paper describes a constraint programming (CP) approach (Barták 1999) to solve a scheduling problem with earliness and tardiness costs, which is for distinct due dates NP-complete already on one resource (Baker & Scudder 1990).

This paper focuses on the job shop scheduling problem with earliness and tardiness costs. This problem—introduced in (Beck & Refalo 2001; 2003)—is solved there using hybrid approach based on probe backtrack search (El Sakkout & Wallace 2000) with integration of constraint programming and linear programing. This hybrid approach performed significantly better than the generic (naive) CP and MIP algorithms. With another hybrid approach, combining local search and linear programming (Beck & Refalo 2002), results slightly worse than in (Beck & Refalo 2001) were obtained. The large neighborhood search (Danna & Perron 2003) applied to the same earliness tardiness job shop problem outperformed both hybrid approaches of Beck & Refalo.

This paper describes a search procedure for scheduling problems with earliness and tardiness costs which initially tries to assign to variables those values that lead to a solution with minimal cost. It is developed by improving of the search procedure used in (Kelbel & Hanzálek 2006) where constraint programming is applied to an industrial case study on a lacquer production scheduling. While in (Kelbel & Hanzálek 2006) tardy jobs were not allowed, the procedure described in this paper allows both early and tardy jobs, i.e. optimal solutions are not discarded.

The proposed search procedure is tested on a set of randomly generated instances of the job shop scheduling problem with earliness and tardiness costs. It significantly outperforms simple (default) models introduced in (Beck & Refalo 2003), and in average it gives results better than the Unstructured Large Neighborhood Search (Danna & Perron 2003).

## Earliness Tardiness Job Shop Scheduling Problem

The definition of the earliness tardiness job shop scheduling problem (ETJSSP) is based on (Beck & Refalo 2003). We assume a set of jobs $\mathcal{J} = \{J_1, \ldots, J_n\}$ where job $J_j$ consists of a set of tasks $\mathcal{T}_j = \{T_{j,1}, \ldots, T_{j,n_j}\}$. Each task has given processing time $p_{j,i}$, and required dedicated unary resource from a set $\mathcal{R} = \{R_1, \ldots, R_m\}$. Starting time $S_{j,i}$ of a task, and completion time defined as $C_{j,i} = S_{j,i} + p_{j,i}$, determine the result of the scheduling problem. For each job $J_j$ there are precedence relations between tasks $T_i$ and $T_{i+1}$ such that $C_{j,i} \leq S_{j,i+1}$ for all $i = 1, \ldots, n_j - 1$, i.e. $\mathcal{T}_j$, the set of tasks, is ordered.

Concerning earliness and tardiness costs, each job has assigned a due date $d_j$, i.e. the time when the last task of the job should be finished. In general, the due dates are distinct. The cost function of the job $J_j$ is defined as $\alpha_j(d_j - C_{j,n_j})$ for early job and $\beta_j(C_{j,n_j} - d_j)$ for tardy job, where $\alpha_j$ and $\beta_j$ are earliness and tardiness costs of the job per time unit. Taking into account both alternatives, the cost function of the job can be expressed as

$$f_j = \max(\alpha_j(d_j - C_{j,n_j}), \beta_j(C_{j,n_j} - d_j)). \qquad (1)$$

An optimal solution of the ETJSSP is the one with minimal possible sum of costs over all jobs

$$\min \sum_{J_j \in \mathcal{J}} f_j.$$

In this article, a specific ETJSSP will be considered in order to be consistent with the original problem instances (Beck & Refalo 2003). All jobs have the sets of tasks with the same cardinality, which is equal to the number of resources, i.e. $n_j = m$ for all $j$. Each of the $n_j$ tasks of the job is processed on a different resource. Next, the problem has a work flow structure: the set of resources $\mathcal{R}$ is partitioned into two disjunctive sets $\mathcal{R}_1$ and $\mathcal{R}_2$ of about the same cardinality, and the tasks of each job must use all resources from

the first set before any resource from the second set, i.e. task $T_{j,i}$ for all $i = 1, \ldots, |\mathcal{R}_1|$ requires resource from set $\mathcal{R}_1$, and task $T_{j,i}$ for all $i = |\mathcal{R}_1| + 1, \ldots, n_j$ requires resource from set $\mathcal{R}_2$.

## The Model With Search Procedure for ETJSSP

When solving constraint satisfaction problems (Barták 1999), constraint programming systems employ two techniques—constraint propagation and search. The search consists of a search tree construction by a search procedure (called also a labeling procedure) and applying a search strategy (e.g. depth-first search) to explore the tree. The search procedure typically makes decisions about variable selection (i.e. which variable to choose) and about value assignment (i.e. which value from domain to assign to the selected variable).

Our approach to solving ETJSSP is based on usual constraint programming model with a problem specific search procedure. The scheduling problem is modeled directly by using a formulation from the previous section, yet by using higher abstraction objects for scheduling (e.g. tasks and resources) available in ILOG OPL Studio (ILO 2002). The model uses scheduling-specific edge-finding propagation algorithm for disjunctive resource constraints (Carlier & Pinson 1990). In the used CP system we obtained better performance of the computations when the cost function (1) was expressed as $f_j \geq \alpha_j(d_j - C_{j,n_j}) \wedge f_j \geq \beta_j(C_{j,n_j} - d_j)$.

Most of the constraint programming systems have a default search procedure that builds the search tree by assigning the values from domains to variables in increasing order. The idea of our search procedure is based on the fact that only $C_{j,n_j}$, the completion time of the last task of the job, influences the value of the cost function, and that the values of $C_{j,n_j}$ inducing the lowest values of cost functions $f_j$ should be examined first.

The search procedure, inspired by time-directed labeling (Van Hentenryck, Perron, & Puget 2000), is directed by the cost, only once at the beginning of the search (as an initialization of the search tree), however. It is denoted as cost-directed initialization (CDI) and performs as described in Algorithm 1: variables representing completion time $C_{j,n_j}$ are selected in increasing order of the size of their domains, then the value selection is made according to the lowest value possible of the cost function. In the second branch of the search tree, this value is disabled. This is done only once for each task $T_{j,n_j}$, then the search continues with the default search procedure.

Slice Based Search available in (ILO 2002), based on (Beck & Perron 2000), and similar to Limited Discrepancy Search (Harvey & Ginsberg 1995) is used as a search strategy to explore the search tree constructed by the CDI procedure. This is necessary for obtaining good performance, since using depth first search instead, the algorithm was not able to find any solution for about 50% of larger size instances of the ETJSSP.

**Algorithm 1 – CDI search procedure**

1. sort the last tasks of all jobs, $T_{j,n_j}$ for all $j$, according to the nondecreasing domain size of $C_{j,n_j}$
2. for each task from the sorted list from domain of $C_{j,n_j}$ select a value $v_j$ leading to minimal $f_j$ and create two alternatives in the search tree:
   - $C_{j,n_j} = v_j$
   - $C_{j,n_j} \neq v_j$
3. Continue with the default search procedure for all variables

## Experimental Results

The proposed algorithm CDI was tested against two simple generic models introduced in (Beck & Refalo 2003), a mixed integer programming model with disjunctive formulation of the problem (MIP), and a constraint programming model with *SetTimes* heuristic as a search procedure and depth-first search as a search strategy (ST). The third model used for performance comparison is the Unstructured Large Neighborhood Search (uLNS) (Danna & Perron 2003) by enabling Relaxation Induced Neighborhood Search (RINS) via `IloCplex::MIPEmphasis=4` switch in Cplex 9.1 (Danna, Rothberg, & Le Pape 2005; ILO 2005), while using the same MIP model as in (Beck & Refalo 2003). The hybrid algorithm from (Beck & Refalo 2003) was not used due to its implementation complicacy.

Benchmarks are randomly generated instances of the ETJSSP according to Section 6.1 in (Beck & Refalo 2003). The problem instances have a work flow structure. Processing times of tasks are uniformly drawn from the interval $[1, 99]$. Considering the lower bound $tlb$ of the makespan of the job shop according to (Taillard 1993), and a parameter called looseness factor $lf$, the due date of the job was uniformly drawn from the interval $[0.75 \cdot tlb \cdot lf, 1.25 \cdot tlb \cdot lf]$. The job shops were generated for three $n \times m$ sizes, $10 \times 10$, $15 \times 10$, and $20 \times 10$, and for $lf \in \{1.0, 1.3, 1.5\}$. Twenty instances were generated for each $lf$—size combination.

The tests were executed using ILOG OPL Studio 3.6 with ILOG Solver and Scheduler for the CP models, and ILOG Cplex 9.1 for the MIP models, all running on a PC with CPU AMD Opteron 248 at 2.2 GHz with 4 GB of RAM. The time limit for each test was 600 s, after which the execution of the test computation was stopped, and the best solution so far was returned.

Table 1 shows the average ratio of the costs of the best solutions obtained by the MIP, uLNS, and ST to the best solutions obtained by CDI, for all types of instances.

In Tables 2 and 3 the ST algorithm will not be included due to its poor performance. Table 2 shows the number of instances solved to optimality within 600 s time limit, and also the number of instances, for which the algorithm proved the optimality of the solution. The CDI usually needed less time than the MIP or uLNS to find a solution with optimal cost, but in many cases it was not proven as an optimum in given time or memory limit. In Table 2 a solution found by the CDI model was considered as the optimal solution when

| size | $10 \times 10$ | | | $15 \times 10$ | | | $20 \times 10$ | | |
|---|---|---|---|---|---|---|---|---|---|
| $lf$ | MIP/CDI | uLNS/CDI | ST/CDI | MIP/CDI | uLNS/CDI | ST/CDI | MIP/CDI | uLNS/CDI | ST/CDI |
| 1.0 | 1.8 | 1.2 | 2.6 | 4.7 | 3.1 | 6.2 | 5.3 | 4.9 | 6.7 |
| 1.3 | 4.8 | 1.8 | 9.2 | 18.4 | 5.3 | 28.3 | 14.0 | 14.3 | 25.8 |
| 1.5 | 3.8 | 2.1 | 8.1 | 7.9 | 1.9 | 37.9 | 5.5 | 5.7 | 50.6 |

Table 1: Average ratio for the best values of cost functions of solutions found within 600 s

the value of the objective function was equal to the one of the proven optimal solution found by the MIP models or to a lower bound found by the MIP.

Table 3 is inspired by (Beck & Refalo 2002). For each problem instance, the lowest cost obtained by any of the used algorithms is selected. Then, Table 3 contains the number of instances for which the algorithm found the solution with the best cost, i.e. equal to the lowest cost, and the number of solutions with uniquely best cost, i.e. if no other algorithm has found solution with the same or lower cost.

## Conclusion and Future Work

We have shown an algorithm called cost-directed initialization (CDI) designed to solve the earliness-tardiness scheduling problem. The algorithm was compared to other algorithms MIP, uLNS, and ST, on randomly generated earliness tardiness job shop benchmarks. The CDI was able to find within 600 s a solution that is usually better than the one found by any of the MIP, uLNS, or ST. With respect to the best obtained value of the cost function, the CDI algorithm performed better than the other algorithms. However, the weak point of the CDI is that the optimum, even if it is found, is usually not proved.

Since the CDI search procedure does not exploit the structure of the job shop problem, it is possible to apply it on other earliness/tardiness problems but the results may vary. Revisiting the lacquer production scheduling problem (Kelbel & Hanzálek 2006) with the CDI, the solution of the case study was further improved from the cost 886,535 to 777,249 due to the allowance of tardy jobs.

The earliness tardiness job shop scheduling problem, as considered in this paper, does not fully correspond to real production, since only the last tasks of jobs have direct impact on the cost of the schedule. If there is enough time, i.e. the looseness factor is big, there can be quite a big delay between the tasks of the same job, and so a storage would be needed also during the production, but at no cost (since no such cost is defined). So the payed storage of the final product can be replaced by the free storage during the production.

There are some approaches to making formulation of this problem closer to real life. Either by assignment of the due date, earliness cost, and tardiness cost to all task (Baptiste, Flamini, & Sourd To appear in 2008), or by introduction of buffers with limited capacity that are used during the production (Brucker *et al.* 2006).

The approach with limited buffers is also used in the formulation of the lacquer production scheduling (Kelbel & Hanzálek 2006) where each job needs a limited buffer (mixing vessel) during the whole time of its execution.

In future, we would like to focus on the formulation and solution of the job shop problems with earliness and tardiness costs and with generic limited buffers.


### Acknowledgement

The work described in this paper was supported by the Czech Ministry of Education under Project 1M0567. Also, we would like to thank the anonymous reviewers for useful comments.


## References


Baker, K. R., and Scudder, G. D. 1990. Sequencing with earliness and tardiness penalties: A review. *Operations Research* 38(1):22–36.

Baptiste, P.; Flamini, M.; and Sourd, F. To appear in 2008. Lagrangian bounds for just-in-time job-shop scheduling. *Computers & Operations Research* 35(3):906–915.

Barták, R. 1999. Constraint programming – what is behind? In *Proc. of CPDC99 Workshop*.

Beck, J. C., and Perron, L. 2000. Discrepancy-bounded depth first search. In *Second International Workshop on Integration of AI and OR Technologies for Combinatorial Optimization Problems (CP-AI-OR'00)*.

Beck, J. C., and Refalo, P. 2001. A hybrid approach to scheduling with earliness and tardiness costs. In *Third International Workshop on Integration of AI and OR Techniques (CP-AI-OR'01)*.

Beck, J. C., and Refalo, P. 2002. Combining local search and linear programming to solve earliness/tardiness scheduling problems. In *Fourth International Workshop on Integration of AI and OR Techniques (CP-AI-OR'02)*.

Beck, J. C., and Refalo, P. 2003. A hybrid approach to scheduling with earliness and tardiness costs. *Annals of Operations Research* 118(1–4):49–71.

Brucker, P.; Heitmann, S.; Hurink, J.; and Nieberg, T. 2006. Job-shop scheduling with limited capacity buffers. *OR Spectrum* 28(2):151–176.

Carlier, J., and Pinson, E. 1990. A practical use of jackson's pre-emptive schedule for solving the job-shop problem. *Annals of Operations Research* 26:269–287.

Danna, E., and Perron, L. 2003. Structured vs. unstructured large neighborhood search: A case study on job-shop

| size | $10 \times 10$ | | | | | | $15 \times 10$ | | | | | | $20 \times 10$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $lf$ | MIP | | uLNS | | CDI | | MIP | | uLNS | | CDI | | MIP | | uLNS | | CDI | |
| | F | P | F | P | F | P | F | P | F | P | F | P | F | P | F | P | F | P |
| 1.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1.3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 2 | 2 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1.5 | 7 | 7 | 9 | 9 | 10 | 4 | 5 | 5 | 9 | 9 | 12 | 3 | 3 | 3 | 4 | 4 | 5 | 3 |

Table 2: Number of optimal solutions (F)ound and (P)roven within 600 s

| size | $10 \times 10$ | | | | | | $15 \times 10$ | | | | | | $20 \times 10$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $lf$ | MIP | | uLNS | | CDI | | MIP | | uLNS | | CDI | | MIP | | uLNS | | CDI | |
| | B | U | B | U | B | U | B | U | B | U | B | U | B | U | B | U | B | U |
| 1.0 | 0 | 0 | 5 | 5 | 15 | 15 | 0 | 0 | 0 | 0 | 20 | 20 | 0 | 0 | 0 | 0 | 20 | 20 |
| 1.3 | 0 | 0 | 2 | 2 | 18 | 18 | 1 | 0 | 2 | 0 | 20 | 18 | 1 | 1 | 1 | 1 | 18 | 18 |
| 1.5 | 8 | 0 | 12 | 0 | 20 | 8 | 5 | 0 | 14 | 3 | 16 | 6 | 3 | 0 | 7 | 4 | 16 | 12 |

Table 3: Number of (B)est and (U)niquely best solutions found within 600 s


scheduling problems with earliness and tardiness costs. In *Ninth International Conference on Principles and Practice of Constraint Programming*, 817–821.

Danna, E.; Rothberg, E.; and Le Pape, C. 2005. Exploring relaxation induced neighborhoods to improve MIP solution. *Mathematical Programming* 102(1):71–90.

El Sakkout, H., and Wallace, M. 2000. Probe backtrack search for minimal perturbation in dynamic scheduling. *Constraints* 5(4):359–388.

Harvey, W. D., and Ginsberg, M. L. 1995. Limited discrepancy search. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, 607–615.

ILOG S.A. 2002. *ILOG OPL Studio 3.6 Language Manual*.

ILOG S.A. 2005. *ILOG Cplex 9.1 User's Manual*.

Kelbel, J., and Hanzálek, Z. 2006. A case study on earliness/tardiness scheduling by constraint programming. In *Proceedings of the CP 2006 Doctoral Programme*, 108–113.

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

Van Hentenryck, P.; Perron, L.; and Puget, J.-F. 2000. Search and strategies in OPL. *ACM Transactions on Computational Logic* 1(2):285–320.