CONSTRAINT-BASED SCHEDULING WITH TOTAL COMPLETION TIME CRITERION: MODELS AND APPLICATIONS

András Kovács, [*] **J. Christopher Beck** [**]

[*] *Computer and Automation Research Institute*
Hungarian Academy of Sciences
akovacs@sztaki.hu
[**] *Dept. of Mechanical & Industrial Engineering*
University of Toronto
jcb@mie.utoronto.ca

Abstract: Constraint programming (CP) offers flexible and effective tools for modeling combinatorial optimization problems. At the same time, scheduling with sum type optimization criteria, such as total completion time is challenging for CP. In this paper we show how extending a standard CP solver by a global constraint on total completion time can boost the performance of CP on various, seemingly very different optimization problems, including job shop scheduling, scheduling with tool changes, or even container loading.

Keywords: Constraint-based scheduling, global constraint, total completion time, applications

1. INTRODUCTION

Constraint programming (CP) offers flexible and effective tools for modeling combinatorial optimization problems. The modeling capabilities of this declarative programming approach derive from the fact that constraints can be combined arbitrarily, therefore it makes possible expressing a broad scale of requirements in a compact way. CP has been successful in practical applications especially in the domain of scheduling.

Solution methods in CP combine search and inference: inference methods restrict the search space to regions where there is a chance of finding high quality solutions. The most important inference method in CP is constraint propagation. There is an algorithm, a so-called propagator attached to each constraint, which removes those values from the variable domains that cannot take part in any feasible solution. A brief introduction to solution methods in constraint-based scheduling is given in (Barták, 2003), while an in-depth presentation of the field can be found in (Baptiste et al., 2001).

A major challenge to constraint-based scheduling is that its computational efficiency depends strongly on the choice of optimization criterion. While it is highly efficient with maximum type objective functions such as makespan, sum type objective functions such as total weighted completion time are more challenging. The reason is that classical CP-based approaches are unable to achieve strong back propagation from these objective functions, i.e., remove the values from the variable domains that are inconsistent with the actual bound on the criterion. Without back propagation, the full search space will need to be explored, suggesting that CP will not result in better performance than any other search technique. One way of tackling this challenge is designing efficient optimization-oriented global constraints to achieve a high degree of back propagation, see, e.g. (Focacci et al., 2002). The COMPLETION constraint is a recently proposed global constraint to propagate the total weighted completion time of activities on a single unary capacity resource (Kovács and Beck, 2007b).

In this paper we present how the extension of a standard CP solver with the COMPLETION constraint can boost its computational efficiency in various, seemingly very different application domains. The problems discussed in this paper are job shop scheduling, scheduling with tool changes, and container loading. In the next section, we briefly present the COMPLETION constraint and the associated propagation algorithm. Then, we consider each of the application domains in separate sections. Finally, conclusions are drawn.

2. A CONSTRAINT FOR TOTAL WEIGHTED COMPLETION TIME

The classical way of representing sum type objective functions in constraint-based scheduling is posting a sum constraint on the end times of the activities. In the case of total weighted completion time, it takes the form $K = \sum_i w_i C_i$, where w_i stands for the weight of activity A_i, and C_i for its end (completion) time. In addition, we use S_i and p_i to denote the start time and duration of A_i.

However, the sum constraint, ignoring the fact that subsets of the activities require the same resource, assumes that all of them can start at their earliest start times. This often leads to very loose lower bounds on K, and weak back propagation. In order to achieve tight lower bounds on K and strong back propagation to the start time variables S_i, the COMPLETION constraint has been introduced in (Kovács and Beck, 2007b) to propagate the total weighted completion time of a set of activities that require the same unary capacity resource. Formally, it is defined as

$$\text{COMPLETION}(\\ [S_1, ..., S_n], [p_1, ..., p_n], [w_1, ..., w_n], K)$$

and enforces $K = \sum_i w_i(S_i + p_i)$. The complete propagation of the relation described by this constraint – technically speaking, achieving generalized bounds consistency – involves solving several single machine scheduling problems with release times and deadlines and an upper bound on the total weighted completion time, formally denoted as $1|r_i, d_i| \sum w_i C_i$. This problem is NP-hard, hence, cannot be solved efficiently each time the COMPLETION constraint has to be propagated. Instead, our propagation algorithm filters domains with respect to the following *relaxation* of the above problem.

The *preemptive mean busy time* relaxation (Goemans et al., 2002), denoted by $1|r_i, pmtn| \sum w_i M_i$, involves scheduling preemptive activities on a single machine with release times respected, but deadlines disregarded. It minimizes the total weighted mean busy times M_i of the activities, where M_i is the average point in time at which the machine is busy processing A_i. This is easily calculated by finding the mean of each time point at which activity A_i is executed. This relaxed problem can be solved to optimality in $O(n \log n)$ time.

The COMPLETION constraint filters the domains of the start time variables by computing the cost of the optimal preemptive mean-busy time relaxation *for each activity* A_i and *each possible start time* t of activity A_i, with the added constraint that activity A_i must start at time t. If the cost of the relaxation is greater than the current upper bound, then t is removed from the domain of S_i. Clearly, the naive computation of all these relaxed solutions would be computationally intractable. The main contribution of (Kovács and Beck, 2007b) is showing that for each activity it is sufficient to compute relaxed solutions for a limited number of different values of t, and that subsequent relaxed solutions can be computed iteratively by a permutation of the activity fragments in previous solutions. For a detailed presentation of this algorithm and the COMPLETION constraint, in general, readers are referred to the above paper.

3. JOB SHOP SCHEDULING

In an $n \times m$ job shop scheduling problem (JSP) there are n jobs to be scheduled on a set of unary capacity resources. Each job is composed of m completely ordered activities, and each activity requires exclusive use of one resource during its execution. Then, a schedule is looked for such that

- no resource executes more than one activity at a time; and
- each activity starts after the end of its preceding activity in the job-order.

The standard JSP decision problem asks if, for a given makespan, D, all activities can finish by D. This is a well-known NP-complete problem. However, it is not uncommon to solve the optimization version of the JSP with the goal of minimizing some metric, such as makespan or, in the present case, the total weighted completion time of the jobs. More formally, given a set of jobs J with weights $w_j, j \in J$ and job-last activities E_j, our goal is to find a feasible schedule that minimizes $\sum_{j \in J} w_j C_{Ej}$.

3.1 *Modeling the Problem*

Applying the COMPLETION constraint to the JSP is straightforward: the only complication is that CP reasons with activities, while in a JSP, weights and performance measures are related to jobs. Therefore, we need to define a mapping from job weight to activity weight. The obvious approach, which we refer to as *last*, is to assign the job weight to the last activity in each job, and to assign zero weight to all other activities. We then place a COMPLETION constraint on each resource, and the solution cost is computed as the sum of total weighted completion times on individual resources.

The drawback of the *last* approach is that COMPLETION constraint reasons over the interaction among activities on the same resource. This interaction is not captured when the weighted activities are on different resources. Since in JSP benchmarks there can be only one job-last activity on each resource, the COMPLETION constraint with the *last* weight mapping is not able to achieve stronger pruning than the simple weighted sum constraint.

Therefore, we propose a different weight mapping, called *busy*. Before solving, we identify the most loaded resource, i.e., the "busy" resource, by summing the durations of the activities on each resource and selecting the resource with highest sum. The weight of each job is assigned to the last activity of the job that is processed on the busy resource. All other activities have a weight of zero. A single COMPLETION constraint can then be posted on the busy resource. To calculate the total weighted completion time, we need to correct for the fact that the weighted activity is not necessarily the last activity in the job. Formally, as above, let E_j be the last activity in job j and let B_j be the single weighted activity in job j. Our optimization function is then: $K+\sum_{j\in J} w_j(C_{E_j} - C_{B_j})$ where K is the cost variable associated with the COMPLETION constraint.

3.2 *Computational Experiments*

To test the effectiveness of the proposed model, we compared it against the standard CP models of the job shop problem: using the weighted sum (WS) form of the optimization function, and/or the *last* weight allocation. We experimented with two styles of search, chronological backtracking and randomized restart. The models and the algorithms have been implemented in ILOG Solver and Scheduler versions 6.1.

The models were tested on 10×10 JSP problems that have been generated using the problem generator of Watson et al. (1999). Since this generator originally considers the problem of minimizing makespan, we transformed them to total weighted completion time problems by assigning a random weight to each job.

A summary of the experimental results is shown in Fig. 1, where the evolution of the mean relative error (MRE) is plotted over solution time for each of the investigated models. The MRE is calculated with respect to the best known solutions, found either by the algorithms tested here or by variations used in preliminary experiments.

The results illustrate that the model using COMPLETION back propagation and the *busy* weight allocation (COMP-BUSY) significantly outperforms the other three variants, independently of the search strategy applied. A detailed presentation of the results can be found in (Kovács and Beck, 2007a).

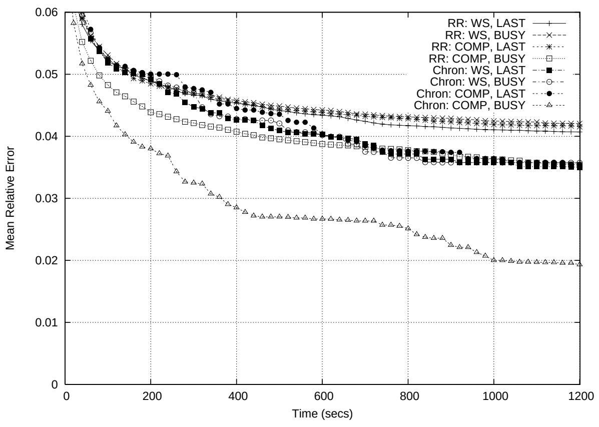

Fig. 1. The mean relative error for different propagation and search techniques. Chron/RR: chronological backtracking or randomized restart search; COMP/WS: COMPLETION or weighted sum back propagation; BUSY/LAST: *busy* or *last* weight allocation.

4. SCHEDULING WITH TOOL CHANGES

In this section we address the problem of single machine scheduling with tool changes, in order to minimize the total completion time of the activities. The regular replacement of the tool is necessary due to wear, which results in a limited, deterministic tool life. We note that this problem is mathematically equivalent to scheduling with periodic preventive maintenance, where there is an upper bound on the continuous running time of the machine. After that, a fixed-duration maintenance activity has to be performed.

This problem has been introduced independently in the periodic maintenance context by Qi et al. (1999) and in the tool changes context by Akturk et al. (2003). The above papers and (Chen, 2006) present various exact optimization approaches to this problem, including different mixed-integer programming (MIP) models and a branch-and-bound search. The performance of several heuristics are also evaluated in the same papers.

4.1 Modeling the Problem

There are various possible ways of modeling the tool changes with constraints. In a recent paper (Kovács and Beck, 2007c) we proposed a so-called machine-time representation, which considers only the active periods of the machine. It exploits that the optimal solution is a no-wait schedule, and contracts each tool change into a single point in time, as shown in Fig. 2. Then, a solution corresponds to a sequencing of the activities, and instantaneous tool changes between them. The machine-time representation implies a natural decomposition of the objective function to the total completion time assuming zero tool change times, and a component representing the effect of positive tool change times.

The model introduced in (Kovács and Beck, 2007c) also contains constraints describing the various dominance rules known for the problem. The joint application of efficient global constraints and dominance rules in the model leads to a compact model and efficient solution process at the same time. This model can be easily adapted to other regular optimization criteria, such as minimizing makespan, or maximum or total tardiness. However, it seems to be somewhat more complicated to extend it to multiple-machine problems with precedence constraints between machines, because the time scales would differ machine by machine.

4.2 Computational Experiments

We ran computational experiments on problem instances taken from (Qi et al., 1999) and (Akturk et al., 2003). Instances with different sizes and characteristics, such as tool lives and tool change times were used. Below we compare the performance of the proposed model with the COMPLETION global constraint to the same model with SUM back propagation. Each row of Table 1 displays cumulated results for 40 instances with a given number of activities, n. Columns *Solved* and *Time* contain the percentage of instances solved to proved optimality and average search time in seconds, including finding the proof of optimality.

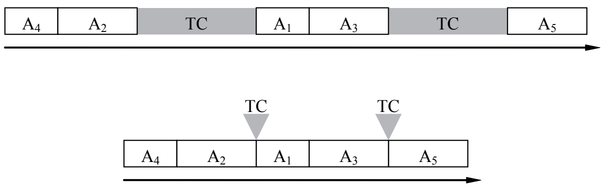

Fig. 2. Wall clock time and machine time representation of the same schedule with tool changes.

The results show that the COMPLETION model outperforms the SUM model even on small instances, and it also scales much better with n. Compared to previous exact optimization approaches, proposed constraint model solved larger instances, significantly faster. The results are presented in detail in (Kovács and Beck, 2007c).

n	SUM		COMPLETION	
	Solved	Time	Solved	Time
15	100.0%	4.0	100.0%	0.0
20	85.0%	951.8	100.0%	0.9
25	7.5%	3453.8	100.0%	19.8
30	-	-	97.5%	126.8
35	-	-	90.0%	657.0
40	-	-	67.5%	1358.0

Table 1. Experimental results on the tool changes problem, for models using SUM and COMPLETION back propagation.

5. CONTAINER LOADING

The container loading problem involves the placement of 3-dimensional items in a container. While the core of the problem corresponds to bin packing with the objective of high volumetric utilization, in practical applications there are various further requirements towards the placement of the items. These include stacking conditions, cargo stability, visibility and accessibility considerations. The rich set of requirements makes CP an attractive modeling approach in this domain. Among the additional requirements, Davies and Bischoff (1999) highlight the importance of the weight distribution of the loaded container, focusing on the location of the center of gravity (COG). The exact requirements depend on the specific application, especially on the means of transport. For example, in aircraft loading, or when loading a container that will be lifted by a crane, the COG of the cargo has to be located in the center of the container. Since the length of the container (or the aircraft hull) is much greater than its width or height, the longitudinal balance is the most important issue. In contrast, in road transport, it is often preferred to have the COG above the axes of the vehicle.

Davies and Bischoff (1999) proposed a loading heuristic that achieves high space utilization combined with an even weight distribution, and claim that the latter leads to a COG located near to the center of the container. Wodziak and Fadel (1994) apply a genetic algorithm to minimize the

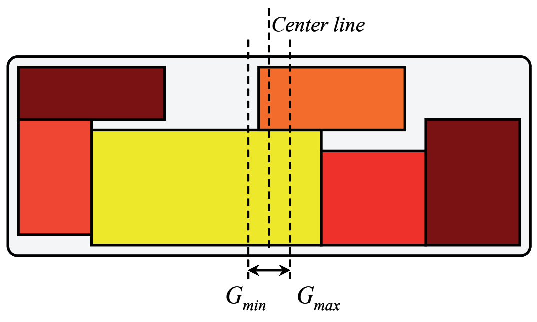

Fig. 3. Sample solution of a container loading problem. The center of gravity has to be located between G_{min} and G_{max}.

distance of the COG from the desired location in one, two, and two and a half dimensions. Various authors have focused on the longitudinal balance, and approached the problem from a one-dimensional perspective, see, e.g. (Mathur, 1998). Fasano (2004) proposed a MIP approach to solving the 3D single bin packing problem with orthogonal rotation allowed and the center of gravity location constrained in all three dimensions.

Below we address the problem of loading box-shaped items into a rectangular container, with rotation disallowed. For simplicity, we present our results on the two-dimensional variant of the problem, and focus on the longitudinal balance. However, it is straightforward to extend the model to k dimensions and constraints on the COG location in all dimensions.

5.1 Modeling the Problem

Let us start by illustrating the relation of container loading to scheduling with total weighted completion time criterion on Fig. 3. The figure can be seen both as the placement of boxes in a container and the Gantt chart representation of a schedule on a discrete resource. The container corresponds to the hull of the schedule, defined by the scheduling horizon (horizontal axis) and the resource capacity (vertical axis). Boxes correspond to activities, with box length standing for activity duration and box width for the resource requirement of the activity. The physical weight of the box corresponds to the weight assigned to activity A_i.

Then, assuming homogeneous boxes, the horizontal coordinate of the COG of box B_i equals $\frac{S_i+C_i}{2}$ in the schedule. This implies that the COG of the complete cargo can be described as

$$\frac{\sum_i w_i \frac{S_i+C_i}{2}}{\sum_i w_i},$$

which in turn can be rewritten to

Container loading	Scheduling
Container length	Scheduling horizon
Container width	Resource capacity
Box length	Activity duration
Box width	Activity's resource req.
COG location	Average weighted completion time + constant bias

Table 2. Corresponding notions in container loading and scheduling.

$$\frac{1}{\sum_i w_i}\left(\sum_i w_i C_i - \sum_i \frac{w_i p_i}{2}\right).$$

Hence, if the total weighted completion time in the schedule is denoted by K, then the COG location equals $G = c_1(K - c_2)$, where $c_1 = \frac{1}{\sum_i w_i}$ and $c_2 = \sum_i \frac{w_i p_i}{2}$ are two constants. The corresponding notions in container loading and scheduling are summarized in Table 2.

We note that the difference between scheduling and container loading is that in scheduling it is allowed to exchange the load of resource units arbitrarily. In container loading, it would mean cutting the loading plan to horizontal stripes and shuffling those stripes, which is infeasible. Hence, discrete resource scheduling with total weighted completion time criterion is a relaxation of container loading with constrained COG location.

In the CP model of the problem, the variables are the horizontal and vertical location of the boxes, and the location of the COG. There is a non-overlapping constraint posted on the boxes. We note that the non-overlapping constraint also makes use of the scheduling relaxation (Clautiaux et al., 2008). For propagating the COG location we use the extension of the COMPLETION constraint to discrete resources presented in (Kovács and Beck, 2007a). We investigate problems where the COG has to be located within a certain range of the center line of the container, between G_{min} and G_{max}.

5.2 Computational Experiments

We compared the performance of this COMPLETION model to the naive model where the COG location was propagated by a weighted sum constraint, called the SUM model. For solving the problem, we implemented different search strategies. The search decisions made by each of the strategies concern a subset of the following: absolute position of a box (AP), relative position of two boxes (RP), and overlap relation of two boxes, i.e., deciding whether two boxes overlap in a given dimension (OR). For the COMPLETION model, the pure AP strategy proved to be the most efficient, and therefore we present experimental results for this strategy. We note that

SUM model performed better with a combined OR+RP+AP strategy than with the AP strategy, but also that combination was outperformed by the COMPLETION model with AP.

Each row of Table 3 contains results for a given number of boxes, n. Columns *Solved* and *Time* display the percentage of solved instances and the average search time. We note that the total number of instances generated for each value of n was 80, with different container sizes. However, the actual number of instances experimented is somewhat lower, because in some problems the container was overloaded.

n	SUM		COMPLETION	
	Solved	Time	Solved	Time
10	100.0%	0.0	100.0%	0.0
15	98.0%	32.9	100.0%	0.2
20	97.6%	121.4	97.6%	29.2
25	68.6%	425.5	100.0%	13.1
30	33.3%	868.6	90.0%	147.3
35	28.6%	872.2	92.9%	178.3
40	31.6%	889.2	78.9%	335.4
45	8.3%	1169.7	66.7%	531.4

Table 3. Experimental results on the container loading problem, for the SUM and the COMPLETION models.

6. CONCLUSION

We demonstrated how the extension of a standard CP solver by a novel global constraint can boost the performance of the solver in various application domains. Namely, we introduced a global constraint for the total weighted completion time of activities that require the same resource, and applied this constraint to three different problems: job shop scheduling, scheduling with tool changes, and container loading with a constraint on the location of the center of gravity. In all of these applications the new approach outperformed the classical CP models, while in single-machine scheduling with tool changes it outstripped all previous exact optimization approaches.

ACKNOWLEDGMENTS

A. Kovács was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences and by the NKFP grant 2/010/2004.

REFERENCES

M. S. Akturk, J. B. Ghosh, and E. D. Gunes. Scheduling with tool changes to minimize total completion time: A study of heuristics and their performance. *Naval Research Logistics*, 50:15–30, 2003.

P. Baptiste, C. Le Pape, and W. Nuijten. *Constraint-based Scheduling.* Kluwer Academic Publishers, 2001.

R. Barták. Constraint-based scheduling: An introduction for newcomers. In *Intelligent Manufacturing Systems 2003*, pages 69–74, 2003.

J-S. Chen. Single-machine scheduling with flexible and periodic maintenance. *Journal of the Operational Research Society*, 57:703–710, 2006.

F. Clautiaux, A. Jouglet, J. Carlier, and A. Moukrim. A new constraint programming approach for the orthogonal packing problem. *Computers & Operations Research*, 35(3):944–959, 2008.

A.P. Davies and E.E. Bischoff. Weight distribution considerations in container loading. *European Journal of Operational Research*, 114:509–527, 1999.

G. Fasano. A MIP approach for some practical packing problems: Balancing constraints and tetris-like items. *4OR: A Quarterly Journal of Operations Research*, 2(2):161–174, 2004.

F. Focacci, A. Lodi, and M. Milano. Optimization-oriented global constraints. *Constraints*, 7(3–4): 351–365, 2002.

M. X. Goemans, M. Queyranne, A. S. Schulz, M. Skutella, and Y. Wang. Single machine scheduling with release dates. *SIAM Journal on Discrete Mathematics*, 15(2):165–192, 2002.

A. Kovács and J. C. Beck. Extensions of the completion constraint. In *Proc. of the CP/ICAPS Joint Workshop on Constraint Satisfaction for Planning and Scheduling Problems*, 2007a.

A. Kovács and J. C. Beck. A global constraint for total weighted completion time. In *Proceedings of CPAIOR'07, 4th Int. Conf. on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (LNCS 4510)*, pages 112–126, 2007b.

A. Kovács and J. C. Beck. Single-machine scheduling with tool changes: A constraint-based approach. In *PlanSIG 2007, the 26th Workshop of the UK Planning and Scheduling Special Interest Group (submitted)*, 2007c.

K. Mathur. An integer-programming-based heuristic for the balanced loading problem. *Operations Research Letters*, 22(1):19–25, 1998.

X. Qi, T. Chen, and F. Tu. Scheduling the maintenance on a single machine. *Journal of the Operational Research Society*, 50:1071–1078, 1999.

J.P. Watson, L. Barbulescu, A.E. Howe, and L.D. 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.

J.R. Wodziak and G.M. Fadel. Packing and optimizing the center of gravity location using a genetic algorithm, 1994. Unpublished manuscript.