Single-machine Scheduling with Tool Changes: A Constraint-based Approach

András Kovács
Computer and Automation Research Institute
Hungarian Academy of Sciences
akovacs@sztaki.hu

J. Christopher Beck
Dept. of Mechanical & Industrial Engineering
University of Toronto
jcb@mie.utoronto.ca

Abstract

The paper addresses the scheduling of a single machine with tool changes in order to minimize total completion time. A constraint-based model is proposed that makes use of global constraints and also incorporates various dominance rules. With these techniques, our constraint-based approach outperforms previous exact solution methods.

Introduction

This paper addresses the problem of scheduling a single machine 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.

Our main intention is to demonstrate the applicability of constraint programming (CP) to an optimization problem that requires complex reasoning with constraints on sum-type expressions, a field were CP is generally thought to be in handicap. We show that indeed, when appropriate global constraints are available to deal with such expressions, CP outperforms other exact optimization techniques. In particular, we would like to illustrate the efficiency of the global COMPLETION constraint (Kovács & Beck 2007), which has been proposed recently for propagating the total weighted completion time of activities on a single unary resource.

For this purpose, we define a constraint model of the scheduling problem. The model makes use of global constraints, and also incorporates various dominance properties described as constraints. A simple branch and bound search is used for solving the problem. We show in computational experiments that the proposed approach can outperform all previous exact optimization methods known for this problem.

The paper is organized as follows. After reviewing the related literature, we give a formal definition of the problem and outline some of its basic characteristics. Then, we propose a constraint-based model of the problem. The algorithms used for propagating the global constraints that are crucial for the performance of our solver are presented. Afterwards, the branch and bound search procedure used is introduced. Finally, experimental results are presented and conclusions are drawn.

Related Work

The problem studied in this paper has been introduced independently in the periodic maintenance context by Qi, Chen, & Tu (1999) and in the tool changes context by Akturk, Ghosh, & Gunes (2003). Its practical relevance is underlined in (Gray, Seidmann, & Stecke 1993), where it is pointed out that in many industries tool change induced by wear is ten times more frequent than change due to the different requirements of subsequent activities. Also, in some industries, e.g. in metal working, tool change times can dominate actual processing times (Tang & Denardo 1988).

Akturk, Ghosh, & Gunes (2003) proposed a mixed-integer programming (MIP) approach and compared the performance of various heuristics on this problem. The basic properties of the scheduling problem have been analyzed and the performance of the Shortest Processing Time (SPT) schedules evaluated in (Akturk, Ghosh, & Gunes 2004). Three different heuristics have been analyzed and a branch and bound algorithm proposed by Qi, Chen, & Tu (1999). The performance of four different MIP models have been compared in (Chen 2006a).

The same problem has been considered with different objective criteria, including makespan (Chen 2007b; Ji, He, & Cheng 2007), maximum tardiness (Liao & Chen 2003), and total tardiness (Chen 2007a). In (Akturk, Ghosh, & Kayan 2007), the model is extended to controllable activity durations, where there are several execution modes available for each activity to balance between manufacturing speed and tool wear. The basic model with several tool types has been investigated by Karakayalı & Azizoğlu (2006). A slightly different problem, in which maintenance periods are strict, i.e. the machine has to wait idle if activities complete earlier than the end of the period, has been investigated in (Chen 2006b).

A brief introduction to constraint-based scheduling is given in (Barták 2003), while an in-depth presentation of the modeling and solution techniques can be found in (Baptiste, Le Pape, & Nuijten 2001).

Problem Definition and Notation

There are n non-preemptive activities A_i to be scheduled on a single machine. Activities are characterized by their durations p_i, and are available from time 0. Processing the activities requires a type of tool that is available in an unlimited number, but has a limited tool life, TL. Worn tools can be replaced with a new one, but only without interrupting activities. This change requires TC time. It is assumed that $\forall i\ p_i \leq TL$, because otherwise the problem would have no solution. The objective is to determine the start times S_i of the activities and start times t_j of tool changes such that the total completion time of the activities is minimal.

Constraint programming uses inference during search on the current domains of the variables. The minimum and maximum values in the current domain of a variable X will be denoted by $\check{X}$ and $\hat{X}$, respectively. Hence, $\check{S}_i$ will stand for the earliest start time of activity A_i, and $\hat{C}_i$ for its latest finish time.

The above parameters and the additional notation used in the paper is summarized in Fig. 1. We assume that all data are integral. A sample schedule is presented in Fig. 2.

n	- Number of activities
p_i	- Duration of activity A_i
p_{max}	- Maximum duration of activities A_i
TL	- Tool life
TC	- Tool change time
S_i	- Start time of activity A_i
C_i	- End (completion) time of activity A_i
t_j	- (Start) time of the jth tool change
a_j	- Number of activities processed after the jth tool change
b_j	- Number of activities processed before the jth tool change
$\check{X}$	- Minimum value in the domain of variable X
$\hat{X}$	- Maximum value in the domain of variable X

Figure 1: Notation

Basic Properties

The single-machine scheduling problem with tool changes, denoted as $1|tool-changes|\sum_i C_i$, has been proven to be NP-hard in the strong sense in (Akturk, Ghosh, & Gunes 2004). The same paper and (Qi, Chen, & Tu 1999) investigated properties of optimal solutions. Below we outline these properties, in conjunction with a symmetry breaking rule that can also be exploited to increase the efficiency of solution algorithms.

Property 1 (No-wait schedule) Activities must be scheduled without any waiting time between them, apart from the tool change times.

Property 2 (SPT within tool) Activities executed with the same tool must be sequenced in the SPT order.

Property 3 (Tool utilization) The total duration of activities processed with the jth tool is at least $TL - p_j^{minafter} + 1$, where $p_j^{minafter}$ is the minimal duration of activities processed with tools $j' > j$.

Consequence Every tool, except for the last one, is utilized during at least $U_{min} = TL - p_{max} + 1$ time, where p_{max} is the largest activity duration. Hence, the number of tools required is at most $\lceil \sum_{i=1}^{n} p_i / U_{min} \rceil$.

Property 4 (Activities per tool) The number of activities processed using the jth tool is a non-increasing function of j.

Property 5 (Symmetry breaking) There exists an optimal schedule in which for any two activities A_i and A_j such that $p_i = p_j$ and $i < j$, A_i precedes A_j.

Modeling the Problem

In our constraint model we apply 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 (see Property 1), and contracts each tool change into a single point in time, as shown in Fig. 3. Then, a solution corresponds to a sequencing of the activities, with the last activity ending at $\sum_i p_i$, and instantaneous tool changes between them.

The objective value of a schedule in the machine time representation takes the form

$$\sum_{i=1}^{n} C_i + TC \sum_{j=1}^{m} a_j.$$

Technically it will be easier to work with b_j than with a_j, hence, we rewrite the objective function to the equivalent form

$$\sum_{i=1}^{n} C_i + TC \sum_{j=1}^{m} (n - b_j).$$

We decompose this function to $K_1 = \sum_{i=1}^{n} C_i$ and $K_2 = TC \sum_{j=1}^{m} (n - b_j)$. Note that K_1 corresponds to the total completion time without tool changes, while K_2 represents the effect of introducing tool changes.

The variables in the model are the start times S_i of the activities, the times t_j of the tool changes, and the number of activities processed before the jth tool change, b_j. The two cost components K_1 and K_2 are also handled as model variables. For the sake of brevity,

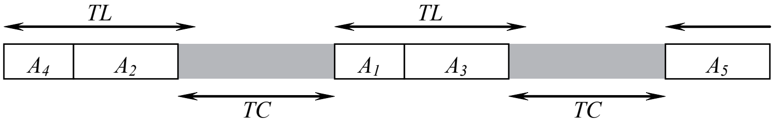

Figure 2: A sample schedule. Wall clock time representation.

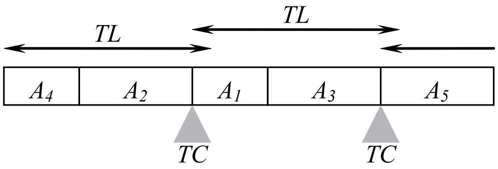

Figure 3: Machine time representation of the sample schedule.

we also use $C_i = S_i + p_i$ to denote the end time of activity A_i.

Then, the problem consists of minimizing $K_1 + K_2$ subject to

(c1) Time window constraints, stating $\forall i: \; S_i \geq 0$ and $C_i \leq \sum_i p_i$;

(c2) Resource capacity constraint: at most one activity can be processed at any point in time;

(c3) Activities are not interrupted by tool changes: $\forall i, j:$ $C_i \leq t_j \; \vee \; S_i \geq t_j$;

(c4) Limited tool life: $\forall j: \; t_{j+1} - t_j \leq TL$;

(c5) Property 3 holds: $\forall j: \; t_{j+1} - t_j \geq TL - p_{max} + 1$;

(c6) Property 4 holds: $\forall j: \; b_j - b_{j-1} \geq b_{j+1} - b_j$;

(c7) Property 5 holds: $\forall i_i, i_2$ such that $i_1 < i_2$ and $p_{i1} = p_{i2}$: $C_{i1} \leq S_{i2}$;

(c8) The total completion time of activities A_i is K_1;

(c9) The number of activities that end before t_j is b_j;

(c10) $K_2 = TC \sum_{j=1}^{m} (n - b_j)$.

Note that while constraints c1-c4 and c8-c10 are fundamental elements of our model, c5-c7 incorporate dominance rules to facilitate stronger pruning of the search space. All the ten constraint can be expressed by languages of common constraint solvers. However, significant improvement in performance can be achieved by applying dedicated global constraints for propagating c8 and c9. We discuss those global constraints in detail in the next section.

Propagation Algorithms for Global Constraints

Below, both for c8 and c9, we first present how the constraint can be expressed in typical constraint languages. Then, we introduce a dedicated global constraint and a corresponding propagation algorithm for either of them, in order to strengthen pruning.

Total Completion Time

The typical way of expressing the total completion time of a set of activities in constraint-based scheduling is posting a sum constraint on their end times: $K = \sum C_i$. However, the sum constraint, ignoring the fact that the activities require the same unary resource, assumes that all of them can start at their earliest start times. This leads to very loose initial lower bounds on K; in the present application $\check{K} = \sum_i p_i$.[1]

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 & Beck 2007) for the total weighted completion time of activities on a 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 $K = \sum_i w_i(S_i + p_i)$. Checking generalized bounds consistency on the constraint requires solving $1|r_i, d_i| \sum w_i C_i$, a single machine scheduling problem with release times and deadlines and upper bound on the total weighted completion time. 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.

[1]The lower bound is a little tighter if symmetry breaking constraints (c7) are present to increase the earliest start times of some activities.

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 relaxed solution is greater than the current upper bound, then t is removed from the domain of S_i. The naive computation of all these relaxed schedules is likely to be too expensive, computationally. The main contribution of (Kovács & Beck 2007) 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.

Number of Activities before a Tool Change

Constraint c8 describes a complex global property of the schedule. Standard CP languages make it possible to express this property with the help of binary logical variables indicating whether a given activity ends before a point in time, i.e.

$$y_{i,j} = \begin{cases} 1 & \text{if } C_i \leq t_j \\ 0 & \text{otherwise.} \end{cases}$$

Then, b_j can be computed as $b_j = \sum_i y_{i,j}$. This representation would be rather inefficient, but implementing a global constraint for this purpose is rather straightforward.

The NBEFORE global constraint states that given activities A_i that have to be executed on the same unary resource, the number of activities that can be completed before time t_j is exactly b_j:

$$\text{NBEFORE}([S_1, ..., S_n], t_j, b_j)$$

The propagation algorithm for this global constraint is presented in Fig. 4. It first determines the set of activities M that *must be* executed before t_j, and the set of activities P that are *possibly* executed before t_j. Computing the minimal (maximal) number of activities scheduled before t_j is performed by sorting P by non-decreasing duration, and then selecting the activities that have the highest (lowest) durations. The algorithm completes by updating $\check{b}_j$, $\hat{b}_j$, and $\check{t}_j$. The time complexity of the propagator is $O(n \log n)$, which is the time needed for sorting P.

We note that it is straightforward to extend this algorithm with propagation from m_j and t_j to S_i, and also to $\hat{t}_j$. This extension has been implemented, but did not achieve additional pruning, and therefore it has been later omitted.

A Branch and Bound Search

We apply a branch and bound search that exploits the dominance properties identified for the problem. It constructs a schedule chronologically, by fixing the start times of activities and the times of tool changes. In each node it selects, according to the SPT rule, the minimal duration unscheduled activity A^* that can be scheduled next. The algorithm first checks if one of the following dominance rules can be applied at this phase of the search.

- If the remaining activities can all be scheduled without any tool changes, then A^* must be scheduled immediately, because all the unscheduled activities must be scheduled according to the SPT rule. See Property 2 and lines 4-5 of the algorithm.
- If A^* cannot be performed before the next tool change, then no unscheduled activities can be performed before the next tool change, since none of them have shorter durations than A^*. Therefore the next tool change must be performed immediately. See Property 1 and lines 6-7 of the algorithm.

If one of the dominance rules can be applied, then the algorithm adds the inferred constraint, which may trigger further propagation, and then reselects A^* w.r.t. the new variable domains. Otherwise, it creates two children of the current search node, according to whether

- A^* is scheduled immediately and the next tool change is performed after (but not necessarily immediately after) A^*; or
- A^* is scheduled after the next tool change.

In the latter case, it also adds the constraint that another activity must be scheduled before the next tool change. Hence, the next tool change must be performed after C_{min}, which is the lowest among the end times of unscheduled activities (see line 9). Note that C_{min} exists because if there is an unscheduled activity (A^*), then there are at least two unscheduled activities.

Also observe that the initial solution found by this branch and bound algorithm is the SPT schedule.

Experimental Result

We ran computational experiments to evaluate the performance of the proposed CP approach from several aspects. We addressed understanding how the COMPLETION and NBEFORE global constraints improve the performance of our model compared to models using only tools of standard CP solvers. We also measured how problem characteristics influence the performance of our approach, and finally, we compared it to previous exact solution methods.

All models and algorithms have been implemented in Ilog Solver and Scheduler version 6.1. The experiments were run on a 2.53 GHz Pentium IV computer with 760 MB of RAM.

Two different problem sets were used for the experiments. The first set was generated as instances in (Qi, Chen, & Tu 1999), the second as in (Akturk, Ghosh, & Gunes 2003). Qi, Chen, & Tu (1999) took activity durations randomly from the interval $[1, 30]$ and fixed

PROCEDURE Propagate()
  $M = \{A_i \mid \hat{S}_i < \check{t}_j\}$
  $P = \{A_i \mid \check{C}_i \le \hat{t}_j\} \setminus M$
  Sort P by non-decreasing duration
  k_{min} = min number of activities in P with total duration $\ge \check{t}_j - \sum_{A_i \in M} p_i$
  k_{max} = max number of activities in P with total duration $\le \hat{t}_j - \sum_{A_i \in M} p_i$
  $\check{b}_j = |M| + k_{min}$
  $\hat{b}_j = |M| + k_{max}$
  $\check{t}_j$ = $\sum_{A_i \in M} p_i$ + total duration of the k_{min} shortest activities in $|P|$

Figure 4: Algorithm for propagating the NBEFORE constraint.

WHILE there are unscheduled activities
  A^* = Unscheduled activity with min $\check{S}_{A^*}$, min p_{A^*}
  T = Earliest tool change time with $\hat{T} > \check{S}_{A^*}$
  IF there is no such T
    ADD $S_{A^*} = \check{S}_{A^*}$ (Property 2)
  ELSE IF $\hat{T} < \check{C}_{A^*}$
    ADD $T = \check{S}_{A^*}$ (Property 1)
  ELSE
    C_{min} = min $\check{C}_i$ of unscheduled activities $A_i \neq A^*$
      BRANCH: - $S_A = \check{S}_A$ and $C_A \le T$
              - $S_A \ge T$ and $T \ge C_{min}$

Figure 5: Pseudo-code of the search algorithm.

the value of TC to 10. The number of activities n has been varied between 15 and 40 in increments of 5, while values of the tool life TL have been taken from $\{50, 60, 70, 80\}$. We generated ten instances with each combination of n and TL, which resulted in 240 problem instances. The time limit for these problems was set to one hour.

In (Akturk, Ghosh, & Gunes 2003), in order to obtain instances with different characteristics, four parameters of the generator were varied, each having a low (L) and a high (H) value. These parameters were the mean and the range of the durations (MD and RD), the tool life (TL), and the tool change time (TC). Generating ten 20-activity instances with each combination of the parameters resulted in $2^4 \cdot 10 = 160$ instances. Since this set contains harder instances, we set the time limit to two hours.

We did not perform comparisons with the MIP models proposed in (Chen 2006a), because that paper presents experimental results only on very easy instances containing few (in most cases only one) tool changes over the scheduling horizon.

Results on Qi's Instances and Comparison to Naive Models

We compared the performance of four different CP models of the problem that represent the two cost components K_1 and K_2 in different ways. K_1 was expressed either by a sum constraint (*Sum*)or by the COMPLETION constraint (*COMPL*), while K_2 was described using binary variables (*Bin*) or the NBEFORE constraint (*NBEF*). Note that the *COMPL/NBEF* is the model proposed in this paper.

The achieved results are displayed in Table 1. Each row contains cumulative results for ten instances with a given value of n and TL. For each of the models, column *Opt* shows the number of instances for which the optimal solution has been found and optimality has been proven, column *Nodes* contains the average number of search nodes, and *Time* the average search time in seconds. *Nodes* and *Time* also contain the effort needed for proving optimality.

The results show that the proposed approach, *COMPL/NBEF* solves instances with up to 30-35 activities to optimality. It outperforms the alternative CP representations that do not benefit from the pruning strength of the COMPLETION and NBEFORE constraints. Instances with a short tool life and hence, many tool changes are more challenging. This is due to the poorer performance of the SPT heuristic, and higher importance of the bin packing aspect of the problem. In contrast, Qi, Chen, & Tu (1999) report that the average solution time of 20-activity instances with their branch and bound approach was in the range of [55.94, 3.57] seconds, depending on the value of TL, and their algorithm could not cope with larger problems.

n	TL	Sum/Bin			COMPL/Bin			Sum/NBEF			COMPL/NBEF		
		Opt	Nodes	Time	Opt	Nodes	Time	Opt	Nodes	Time	Opt	Nodes	Time
15	50	10	36278	10.8	10	877	0.0	10	31134	5.4	10	49	0.0
	60	10	55477	13.6	10	1018	0.2	10	49975	7.7	10	76	0.0
	70	10	18275	3.1	10	358	0.0	10	14357	1.5	10	17	0.0
	80	10	19748	2.9	10	303	0.0	10	15502	1.4	10	19	0.0
20	50	6	5365305	2605.5	10	42853	35.1	8	6579567	1685.3	10	7183	3.7
	60	7	5365603	1778.5	10	19092	16.2	7	7511826	1436.0	10	133	0.0
	70	9	2544734	735.1	10	8051	7.1	9	3119249	558.0	10	84	0.0
	80	10	910496	241.8	10	1957	1.4	10	762404	127.8	10	46	0.0
25	50	0	6282502	3600.0	10	639147	727.3	0	11727713	3600.0	10	99239	78.0
	60	0	9132083	3600.0	10	91385	126.4	0	15404729	3600.0	10	1126	0.4
	70	1	10815570	3587.7	10	83095	104.2	2	16222223	3327.3	10	979	0.2
	80	1	11484097	3358.2	10	91029	122.1	1	16808958	3287.7	10	1082	0.6
30	50	-	-	-	3	2581475	3229.5	-	-	-	9	230088	452.5
	60	-	-	-	4	2093233	2804.0	-	-	-	10	55374	46.9
	70	-	-	-	8	961460	1640.2	-	-	-	10	7877	6.6
	80	-	-	-	10	318435	560.9	-	-	-	10	1721	1.1
35	50	-	-	-	0	3108739	3600.0	-	-	-	7	1724651	2002.6
	60	-	-	-	0	3193284	3600.0	-	-	-	9	355709	449.5
	70	-	-	-	0	2858550	3600.0	-	-	-	10	160239	166.9
	80	-	-	-	2	2000949	3162.0	-	-	-	10	8121	8.9
40	50	-	-	-	-	-	-	-	-	-	1	2371440	3297.7
	60	-	-	-	-	-	-	-	-	-	6	1088871	1597.6
	70	-	-	-	-	-	-	-	-	-	10	279844	393.5
	80	-	-	-	-	-	-	-	-	-	10	85854	143.3

Table 1: Experimental results on instances from (Qi, Chen, & Tu 1999): number of instances where optimality has been proven (Opt), average number of search nodes (Nodes), and average solution time in seconds (Time), for four different CP models. The models use binary variables (*Bin*) or the *NBEFORE* constraint, and a *Sum* or a *COMPLETION* constraint to express the objective function. Dash '-' means that none of the instances with the given n could be solved to optimality.

Results on Akturk's Instances and Effect of Problem Characteristics

Experimental results on the instances from (Akturk, Ghosh, & Gunes 2003) are presented in Table 2. The results on the l.h.s. have been achieved by a naive model with sum back propagation instead of the COMPLETION constraint, the results on the r.h.s. by the complete CP model.

Each row displays data belonging to a given choice of parameters MD, RD, TL, and TC, as shown in the leftmost columns. While the COMPLETION model managed to solve all instances to optimality and also proved optimality, the sum model missed finding the optimum for 2 instances and proving optimality in 5 cases. The COMPLETION model was 10 times faster on average than the sum model.

These results confirm that short tool life implies many tool changes and renders problems more complicated for our model. Low mean duration makes things easier, which is probably due to the higher number of symmetric activities, since these activities can be ordered a priori. Although a low range of durations has a similar effect, it also has a negative impact on the performance of the SPT heuristic, among which the latter seems to be the stronger.

Compared to the MIP approach presented in (Akturk, Ghosh, & Gunes 2003) our CP model solves more instances, and does this more quickly: the MIP model achieved an average solution time of 1904 seconds, it was not able to solve all instances, and for the 15% of the instances it found worse solutions than one of the heuristics.

Conclusion

A constraint-based approach has been presented to single machine scheduling with tool changes. The proposed model outperforms previous exact optimization methods known for this problem. This result is significant especially because the problem requires complex reasoning with sum-type formulas, which does not belong to the traditional strengths of constraint programming. This was made possible by two algorithmic techniques: global constraints and dominance rules. Specifically, we applied the recently introduced COMPLETION constraint to propagate total completion time, and defined a new global constraint, NBEFORE, to compute the number of activities that complete before a given point in time. Furthermore, we could formulate the known dominance properties as constraints in the model.

The introduced model can be easily extended with constraints on the number of tools and with weighted activities. The machine-time representation is appli-

MD	RD	TL	TC	NBEF/Sum				NBEF/COMPL			
				Opt	MRE	Nodes	Time	Opt	MRE	Nodes	Time
L	L	L	L	10	0	1891018	529.9	10	0	38128	23.3
L	L	L	H	10	0	968087	205.9	10	0	102237	52.1
L	L	H	L	10	0	79344	11.9	10	0	237	0.1
L	L	H	H	10	0	12269	1.6	10	0	73	0.0
L	H	L	L	10	0	667659	171.8	10	0	3692	2.3
L	H	L	H	10	0	127866	23.7	10	0	78955	25.7
L	H	H	L	10	0	78775	13.2	10	0	27	0.0
L	H	H	H	10	0	6664	0.7	10	0	29	0.0
H	L	L	L	7	1.71	16430139	3548.8	10	0	1614494	596.4
H	L	L	H	10	0	5606737	1018.0	10	0	47902	25.1
H	L	H	L	10	0	2170750	357.9	10	0	895	0.3
H	L	H	H	10	0	222435	40.6	10	0	9023	3.6
H	H	L	L	8	0	6020041	2102.8	10	0	81249	43.9
H	H	L	H	10	0	186735	35.7	10	0	23214	11.3
H	H	H	L	10	0	86856	12.5	10	0	20	0.0
H	H	H	H	10	0	154639	19.2	10	0	1648	0.8

Table 2: Experimental results on instances from (Akturk, Ghosh, & Gunes 2003), for models using sum and COMPLETION back propagation: number of instances where optimality has been proven (Opt), mean relative error in percents (MRE), average number of search nodes (Nodes), and average solution time in seconds (Time).

cable to solving the same problem with other regular optimization criteria, such as minimizing makespan, or maximum or total tardiness. However, it seems to be impractical to apply this model to multiple-machine problems, because the time scales would differ machine by machine.

Acknowledgments The authors are grateful to the anonymous reviewers for their helpful comments. 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

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

Akturk, M. S.; Ghosh, J. B.; and Gunes, E. D. 2004. Scheduling with tool changes to minimize total completion time: Basic results and SPT performance. *European Journal of Operational Research* 157:784–790.

Akturk, M. S.; Ghosh, J. B.; and Kayan, R. K. 2007. Scheduling with tool changes to minimize total completion time under controllable machining conditions. *Computers and Operations Research* 34:2130–2146.

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

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

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

Chen, W. J. 2006b. Minimizing total flow time in the single-machine scheduling problem with periodic maintenance. *Journal of the Operational Research Society* 57:410–415.

Chen, J.-S. 2007a. Optimization models for the tool change scheduling problem. *Omega* (to appear).

Chen, J.-S. 2007b. Scheduling of nonresumable jobs and flexible maintenance activities on a single machine to minimize makespan. *European Journal of Operational Research* (to appear).

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

Gray, E.; Seidmann, A.; and Stecke, K. E. 1993. A synthesis of decision models for tool management in automated manufacturing. *Management Science* 39:549–567.

Ji, M.; He, Y.; and Cheng, T. C. E. 2007. Single-machine scheduling with periodic maintenance to minimize makespan. *Computers & Operations Research* 34:1764–1770.

Karakayalı, I., and Azizoğlu, M. 2006. Minimizing total flow time on a single flexible machine. *International Journal of Flexible Manufacturing Systems* 18:55–73.

Kovács, A., and Beck, J. C. 2007. 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 Com-*

binatorial Optimization Problems (LNCS 4510), 112–126.

Liao, C. J., and Chen, W. J. 2003. Single-machine scheduling with periodic maintenance and nonresumable jobs. *Computers & Operations Research* 30:1335–1347.

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

Tang, C. S., and Denardo, E. V. 1988. Models arising from a flexible manufacturing machine, Part I: Minimization of the number of tool switches. *Operations Research* 36:767–777.