Constraint-Based Task Scheduling with Sequence Dependent Setup Times, Time Windows and Breaks

Armin Wolf

Fraunhofer FIRST, Kekuléstr. 7, D-12489 Berlin, Germany
armin.wolf@first.fraunhofer.de

Abstract: The work presented in this article generalizes the modeling of task scheduling problems with sequence dependent setup time on the basis of task scheduling on single respectively exclusive resources: Besides working activities also work breaks are respected properly which is formally proven. The additional contribution of this article is an effective pruning algorithm according to the presented modeling approach. Benchmark examinations show that the introduced modeling and pruning approach is comparable with another state-of-the-art constraint-based approach. Within these examination the optimality of a lower bound of one benchmark instance, namely of t2-ps09, is proven – to the best of one's knowledge – the first time.

1 Introduction

Production or maintenance scheduling is still an attractive field for researchers and practitioners. Researchers are interested in better approaches for those in general NP-complete scheduling problems. In *Constraint Programming* (CP) the main focus of research concerning scheduling problems lies on

- (more) sophisticated problem models,
- (more) efficient/effective pruning algorithms,
- (more) efficient (heuristic) search methods.

Practitioners are interested in successful applications of modeling-pruning-search combinations in their problem domains. In production and maintenance the computation of feasible and even good schedules that are fast and easily adaptable in the case of disturbances are addressed. The reasons are manifold:

- meeting delivery dates,
- increased/balanced resource workload,
- reduction of through-put times or door-to-door times,
- reduction of any kinds of costs, etc.

In this article constraint-based scheduling of non-preemptive tasks on exclusive resources (i.e. single machines) with *sequence dependent setup times* and *time windows* is considered. Besides working tasks, work breaks are considered, too. The reason for distinguishing work breaks from working tasks is that breaks do not require any setup time however they have to be performed within time windows, e.g. due to legal requirements. – To the best of one's knowledge, this is the first time that work breaks are considered in this scheduling context.

The presented work mainly focuses on the sophisticated modeling of such scheduling problems in CP and its adequate support using existing pruning algorithms in combination with a new pruning algorithm. The impact of the presented modeling-pruning approach is shown by experimental examinations on job shop scheduling benchmark instances where branch & bound optimization is performed on top of a search method developed and presented for job shop scheduling without any setup times.

An important practical application of scheduling with sequence dependent setup times and time windows is *field workforce scheduling* e.g. of maintenance tasks in telecommunication networks [LVA$^+$03] or water distribution networks [SR08]. There, the objective is the reduction of travel times and finally the reduction of the according costs.

2 Related Work

The literature surveys given in [GPG01, AF08] points out the importance of treating sequence dependent setup times in production scheduling and that such scheduling problem are NP-hard. The reviews show the variety of applications considering sequence dependent setup times: the application areas are ranging from paper and plastics industries over printing industry to textile, pharmaceutical, chemical and metallurgical industries.

Due to its importance in industry (and maybe due to its NP-completeness) scheduling with sequence dependent setup times is (still) a challenging topic in Operations Research (OR), Artificial Intelligence (AI) and Constraint Programming (CP). For instance, the OR optimization approach presented in [BT96] for general shop problems with sequence dependent setup times uses a branch & bound method in particular for job shop scheduling benchmark instances. These instances are examined in the presented work using a similar branch & bound method, too (cf. Sec. 5).

AI approaches are presented for example in [GPG01, GVV08]. While Ant Colony Optimization (ACO) is used in [GPG01], a hybrid approach combing genetic algorithms with local search (GA+LS) is applied in [GVV08] to job shop scheduling problems with sequence dependent setup times.

CP approaches for scheduling with sequence dependent setup times are presented in [ABF04, AF08, Vil02, VB02]. In [ABF04] job-shop scheduling with sequence dependent setup times is considered. There, the lower bounds for the make-span on each machine are computed considering the scheduling on each machine independently. In detail, for each machine a *traveling salesman problem with time windows* (TSPTW) is modeled and solved. However, the pruning algorithms used to solve the TSPTW are not presented/described. Specialized pruning algorithms are presented in [AF08] which seems to be a continuation of the work presented in [ABF04]. The approaches presented in [Vil02, VB02] are only feasible for a rather small number of different setup times, i.e. so called *setup time families*. The reason is that the pruning algorithms require a precomputation of minimal setup times for all non-empty, exponentially many subsets of such families.

The work presented in this article is based on the results in [BL98]. Additionally, time windows as well as breaks are considered and a new pruning algorithm is introduced that reduces runtime significantly. This pruning algorithm is based on *detectable precedences* [Vil04, Vil07] (cf. Sect. 4).

3 Scheduling with Sequence Dependent Setup Times

A formal description of the task scheduling problem is given in this section. It starts with the definition of the items to be scheduled: working tasks and work breaks. Then, the definition of sequence dependent setup times is given. These definitions are completed by two kinds of interrelated scheduling problems which are formally specified. Finally it is shown that both scheduling problems are equivalent with respect to their solutions.

3.1 Tasks and Breaks

The considered scheduling problem consists of a set of *tasks* T, a set of (task) families F indicating the *type* of each task, a set of *breaks* B and an exclusively available resource r. Each task $i \in T$ is a non-preemptive activity defined by its

- *earliest start time* $\mathsf{est}_i \in \mathbb{Z}$,
- *latest start time* $\mathsf{lst}_i \in \mathbb{Z}$,
- *earliest completion time* $\mathsf{ect}_i \in \mathbb{Z}$,
- *latest completion time* $\mathsf{lct}_i \in \mathbb{Z}$,
- fixed (or variable) *duration* $d_i \in \mathbb{N}$,
- fixed *family* $f_i \in F$,
- variable *start time* s_i,
- variable *end time* e_i,

Breaks are non-preemptive activities, too; the only difference between breaks and tasks is that breaks do not belong to a family.

For each task or break its earliest start time and latest completion time determines its *time window* for its processing. More formally, for each task or break $x \in T \cup B$ the following constraints must hold:

$$s_x + d_x = e_x \land s_x \in [\mathsf{est}_x, \mathsf{lest}_x] \land e_x \in [\mathsf{ect}_x, \mathsf{lct}_x] \enspace .$$

For simplicity, the set of tasks and breaks $T \cup B$ is sometimes identified by the integer set $\{1, \ldots, |T|+|B|\}$. Then by convention, the integer set $\{i_1, \ldots, i_{|T|}\}$ identify the tasks and the integer set $\{p_1, \ldots, p_{|B|}\}$ the breaks.

3.2 Sequence Dependent Setup Times

The *sequence dependent setup times* between two families $f, g \in F$ is defined by a *setup time matrix* $A^{F \times F} = (a_{f,g})$ where $a_{f,g} \in \mathbb{N}$ defines the required setup time between a task of family $f \in F$ and a task of family $g \in F$:

- no setup time is assumed between any two tasks of the same family, i.e.:

  $$\forall f \in F : a_{f,f} = 0 \enspace ,$$

- any insertion of another task between any two tasks will not decrease the total setup time (triangle inequality), i.e.:

  $$\forall f, g, h \in F : a_{f,h} \leq a_{f,g} + a_{g,h} \enspace .$$

Optionally the setup times may be symmetric[1], i.e.: $\forall f, g \in F : a_{f,g} = a_{g,f}$.

3.3 Sequence Dependent Setup Times Scheduling Problems

A *sequence dependent setup times scheduling problem* (possibly with breaks) is defined by a quadruple $(T, F, B, A^{F \times F})$, where T is a non-empty set of tasks, F is an according, non-empty set of families, B is a possibly empty set of breaks and $A^{F \times F}$ is a setup time matrix.

A *solution* of such a *sequence dependent setup times scheduling problem* (possibly with breaks) is defined by an admissible value assignment of the start times and end times of the tasks and the breaks (as well as of their durations, if they are variable) such that the following constraints are satisfied:

- for each task or break $x \in T \cup B$ the *start-duration-end* and the *time-window* condition

  $$s_x + d_x = e_x \land s_x \in [\mathsf{est}_x, \mathsf{lst}_x] \land e_x \in [\mathsf{ect}_x, \mathsf{lct}_x] \enspace ,$$

[1] This is not always the case, e.g. if setup times represent travel times. There the travel time from A to B may differ from the travel time for the opposite direction.

- for each pair of different tasks or breaks $x, y \in T \cup B$ with $x \neq y$ the *non-overlapping* condition

  $$e_x \leq s_y \vee e_y \leq s_x \quad ,$$

- for each pair of different tasks $i, j \in T$ with $i \neq j$ in particular the *non-overlapping with setup time* condition

  $$e_i + a_{f_i,f_j} \leq s_j \vee e_j + a_{f_j,f_i} \leq s_i \quad ,$$

- for each pair of different tasks $i, j \in T$ with $i \neq j$ and each break $p \in B$ the *break* condition[2]

  $$(e_i \leq s_p \wedge e_p \leq s_j) \rightarrow (e_i + a_{f_i,f_j} + d_p \leq s_j) \quad .$$

3.4 Multi-Resource Scheduling Problems

The *multi-resource scheduling problem* according to a sequence dependent setup times scheduling problem $(T, F, B, A^{F \times F})$ (possibly with breaks) is defined by the set of tasks $T \cup \bigcup_{g \in F_T} T_g$ where $F_T = \{f_i \mid i \in T\}$ and $T_g = \{i_g \mid i \in T\}$. Each task $i_g \in T_g$ is defined by its fixed (or variable) *duration* $d_{i,g} = d_i$, its variable *start time* $s_{i,g}$ and its variable *end time* $e_{i,g}$ as well as a set of breaks $B \cup \bigcup_{g \in F_T} B_g$ where $B_g = \{p_g \mid p \in B\}$. Each break $p_g \in B_g$ is defined by its fixed (or variable) *duration* $d_{p,g} = d_p$, its variable *start time* $s_{p,g}$ and its variable *end time* $e_{p,g}$.

A *solution* of such a *multi-resource scheduling problem* is defined by admissible values of the start and end times of the tasks and breaks (as well as of their durations, if they are variable) such that the following constraints are satisfied:

- for each task or break $x \in T \cup B$ the *start-duration-end* and *time-window* condition

  $$s_x + d_x = e_x \wedge s_x \in [\mathsf{est}_x, \mathsf{lst}_x] \wedge e_x \in [\mathsf{ect}_x, \mathsf{lct}_x] \quad ,$$

- for each family $g \in F_T$ and each task or break $x_g \in T_g \cup B_g$ the *start-duration-end* condition

  $$s_{x_g} + d_{x_g} = e_{x_g} \quad ,$$

- for each pair of different tasks or breaks $x, y \in T \cup B$ with $x \neq y$ the *non-overlapping* condition

  $$e_x \leq s_y \vee e_y \leq s_x \quad ,$$

[2] This condition ensures that breaks do not occur during the setup times, which are in general working times, too.

- for each family $g \in F_T$ and each pair of different tasks of breaks $x_g, y_g \in T_g \cup B_g$ with $x_g \neq y_g$ the *non-overlapping* condition

$$e_{x,g} \leq s_{y,g} \vee e_{y,g} \leq s_{x,g} \ ,$$

- for each family $g \in F_T$ and each task $i_g \in T_g$ the *offset* condition

$$s_{i,g} = s_i + a_{f_i,g} \wedge e_{i,g} = e_i + a_{f_i,g} \ ,$$

- for each pair of different tasks or breaks $x, y \in T \cup B$ the *common order* condition

$$(e_x \leq s_y) \rightarrow \forall f \in F_T : (e_{x,f} \leq s_{y,f}) \ .$$

It has to be noted that this multi-resource scheduling traces back to the modeling considerations made in [BL98] for scheduling problems with sequence dependent setup times, however, without any work breaks. There, it is shown that these problems are equivalent. This also holds for the considered generalization with work breaks.

3.5 Equivalence of these Scheduling Problems

Lemma 1 *Any solution of a sequence dependent setup times scheduling problem (possibly with breaks) determines a solution of the according multi-resource scheduling problem.*

Proof 1 *Let a solution of a sequence dependent setup times scheduling problem – without loss of generality with breaks – $(T, F, B, A^{F\times F})$ be given. Then, due to* non-overlapping, *the tasks and breaks in $T \cup B$ are linearly ordered. Assuming that the tasks and breaks are sorted according to their scheduled order it holds $e_x \leq s_{x+1}$ for $x = 1, \ldots, |T| + |B| - 1$. In particular for $i = j_1, \ldots, j_{|T|-1}$ it holds*

$$e_i + a_{f_i,f_{i+1}} \leq s_{i+1}$$

under the assumption that the tasks are numbered according to their scheduled order. Due to the fact that the triangle inequality holds, it follows immediately that $a_{f_i,f_{i+1}} + a_{f_{i+1},g} \geq a_{f_i,g}$ – or equivalently $a_{f_i,f_{i+1}} \geq a_{f_i,g} - a_{f_{i+1},g}$ – holds for any family $g \in F_T$. Thus it holds

$$s_{i+1} \geq e_i + a_{f_i,f_{i+1}} \geq e_i + a_{f_i,g} - a_{f_{i+1},g} \ ,$$

which is equivalent to

$$e_{i,g} = e_i + a_{f_i,g} \leq s_{i+1} + a_{f_{i+1},g} \ ,$$

meaning that the solution of the sequence dependent setup times scheduling problem determines a schedule for the tasks of the according multi-resource problem if

$$s_{i,g} = s_i + a_{f_i,g} \quad \textit{and} \quad e_{i,g} = e_i + a_{f_i,g}$$

holds for $i = j_1, \ldots, j_{|T|}$ *and for each family* $g \in F_T$.

Now we show that there are start and end times for the breaks such that the scheduled order is valid for all families.

In the first step let $q_1, \ldots, q_{k_1}$ *be the breaks scheduled before the first task* j_1, *i.e. it holds* $e_{p_0} \leq s_{j_1}$ *for each* $p_0 \in \{q_1, \ldots, q_{k_1}\}$. *Then for any solution of the according multi-resource problem it holds* $e_{p_0} \leq s_{j_1,g}$ *for each family* $g \in F_T$ *and each* $p_0 \in \{q_1, \ldots, q_{k_1}\}$ *because* $s_{j_1,g} = s_{j_1} + a_{f_{j_1},g} \geq s_{j_1}$ *has to be satisfied. Then, it holds* $e_{p_0,g} \leq s_{j_1,g}$ *for each family* $g \in F_T$ *and each* $p_0 \in \{q_1, \ldots, q_{k_1}\}$ *if*

$$s_{p_0,g} = s_{p_0} \quad \text{and} \quad e_{p_0,g} = e_{p_0}$$

is chosen. Furthermore, the considered breaks are ordered for all families in F_T *in the scheduled linear order.*

Now, in the i*-th step* $(1 \leq i < |T|)$ *let* $q_{k_i+1}, \ldots, q_{k_{i+1}}$ *be the breaks scheduled between task* j_i *and task* j_{i+1}*:*

$$e_{j_i} \leq s_{p_i} \wedge e_{p_i} \leq e_{j_{i+1}}$$

holds for each $p_i \in \{q_{k_i+1}, \ldots, q_{k_{i+1}}\}$. *It follows that*

$$e_{j_i,g} \leq s_{p_i,g} \wedge e_{p_i,g} \leq e_{j_{i+1},g}$$

holds for each family $g \in F_T$ *and each* $p_i \in \{q_{k_i+1}, \ldots, q_{k_{i+1}}\}$ *if*

$$s_{p_i,g} = s_{p_i} + a_{f_{j_i},g} \quad \text{and} \quad e_{p_i,g} = e_{p_i} + a_{f_{j_i},g}$$

is chosen. Furthermore, the considered breaks are ordered for all families in F_T *in the scheduled linear order.*

In the last step let $q_{k_{|T|}+1}, \ldots, q_{|B|}$ *be the breaks scheduled after the last task* $j_{|T|}$, *i. e.* $e_{|T|} \leq s_{p_{|T|}}$ *holds for each* $p_{|T|} \in \{q_{k_{|T|}+1}, \ldots, q_{|B|}\}$. *Then for any solution of the according multi-resource problem it holds* $e_{j_{|T|},g} \leq s_{p_{|T|}} + a_{f_{j_{|T|}},g}$ *for each family* $g \in F_T$ *and each* $p_{|T|} \in \{q_{k_{|T|}+1}, \ldots, q_{|B|}\}$ *because* $e_{j_{|T|},g} = e_{j_{|T|}} + a_{f_{j_{|T|}},g}$ *has to be satisfied. Then, it holds* $e_{j_{|T|},g} \leq s_{p_{|T|},g}$ *for each family* $g \in F_T$ *and each* $p_{|T|} \in \{q_{k_{|T|}+1}, \ldots, q_{|B|}\}$ *if*

$$s_{p_{|T|},g} = s_{p_{|T|}} + a_{f_{j_{|T|}},g} \quad \text{and} \quad e_{p_{|T|},g} = e_{p_{|T|}} + a_{f_{j_{|T|}},g}$$

is chosen. Furthermore, the considered breaks are ordered for all families in F_T *in the scheduled linear order.*

All together, the solution of the sequence dependent setup times scheduling problem with breaks determines a solution of the according multi-resource problem. □

Lemma 2 *Let a sequence dependent setup times scheduling problem (possibly with breaks) be given. Then, any solution of the according multi-resource scheduling problem is a solution of the given sequence dependent setup times scheduling problem.*

Proof 2 *Let* $(T, F, B, A^{F\times F})$ *be a considered sequence dependent setup times scheduling problem – without loss of generality with breaks. Further, let a solution of the according multi-resource problem be given. Then, due to* non-overlapping *and the* common-order *condition, the tasks and breaks in* $T \cup B$ *respective* $T_g \cup B_g$ *are linearly ordered. Assuming that the tasks and breaks are sorted according to their scheduled order it holds*

$$e_x \leq s_{x+1} \wedge e_{x,g} \leq s_{x+1,g}$$

for $x = 1, \dots, |T| + |B| - 1$ *and each family* $g \in F_T$*. In particular for* $i = j_1, \dots, j_{|T|-1}$ *and each family* $g \in F_T$ *it holds*

$$e_i \leq s_{i+1} \wedge e_{i,g} \leq s_{i+1,g}$$

under the assumption that the tasks are numbered according to the scheduled order. By definition it holds

$$s_{i,g} = s_i + a_{f_i,g} \wedge e_{i,g} = e_i + a_{f_i,g}$$

for $i = j_1, \dots, j_{|T|}$ *and each family* $g \in F_T$ *and thus*

$$e_i + a_{f_i,g} \leq s_{i+1} + a_{f_{i+1},g}$$

for $i = j_1, \dots, j_{|T|-1}$ *and each family* $g \in F_T$ *or equivalently*

$$e_i + a_{f_i,g} - a_{f_{i+1},g} \leq s_{i+1} \quad .$$

This inequality holds for each $g \in F_T$ *and thus in particular for* $g = f_{i+1}$*. So, due to the fact that* $a_{f_{i+1},f_{i+1}} = 0$ *holds, it follows immediately*

$$e_i + a_{f_i,f_{i+1}} \leq s_{i+1} \quad .$$

Now, it is assumed that $e_i \leq s_p \wedge e_p \leq s_j$ *holds for an arbitrary pair of different tasks* $i \neq j$ *and an arbitrary break* $p \in B$*. Then for each family* $g \in F_T$ *it holds*

$$e_{i,g} + d_p \leq s_p + d_p \leq s_{j,g}$$

because $e_x = s_x + d_x$ *holds for* $x = 1, \dots, |T| + |B|$*. Furthermore, it holds*

$$e_i + a_{f_i,g} + d_p \leq s_j + a_{f_j,g}$$

because $e_{i,g} = e_i + a_{f_i,g} \wedge s_{j,g} = s_j + a_{f_j,g}$ *is satisfied for each* $g \in F_T$ *and thus in particular for* $g = f_j$*. It follows immediately that*

$$e_i + a_{f_i,f_j} + d_p \leq s_j$$

holds because $a_{f_j,f_j} = 0$ *holds by definition.*

This means that the solution of this multi-resource scheduling is a solution of the considered sequence dependent setup times scheduling problem. □

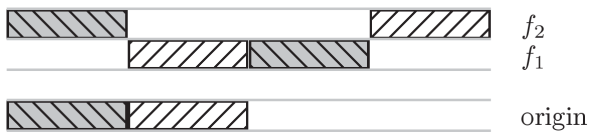
Theorem 1 (Equivalence) *The sequence dependent setup time scheduling problems (possibly with breaks) are equivalent to their according multi-resource scheduling problems with respect to their solutions.*

Proof 3 *The equivalence follows directly from Lemma 1 and Lemma 2.* □

Remark 1 *It is remarkable that the common order over all families in the multi-resource scheduling problem is essential for the equivalence of both problems. As a counter-example consider the sequence dependent setup times scheduling problem with two tasks* 1 *and* 2 *having the setup times*

a_{f_i,f_j}	f_1	f_2
f_1	0	8
f_2	8	0

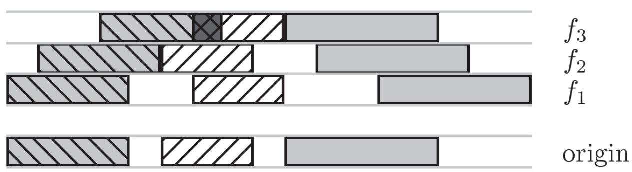
for their families f_1, f_2 *and durations* $d_1 = d_2 = 4$*. Ignoring the required common order in the according multi-resource scheduling problem, there is the solution with* $s_1 = 0, s_2 = 4$ *which is obviously not a solution of the sequence dependent setup times scheduling problem because the setup time* a_{f_1,f_2} *is not respected:*



Remark 2 *It is further remarkable that the triangle equality is essential for the equivalence of both problems: As a counter-example consider the sequence dependent setup times scheduling problem with three tasks* $1, 2, 3$ *having the setup times*

a_{f_i,f_j}	f_1	f_2	f_3
f_1	0	1	3
f_2	1	0	1
f_3	3	1	0

for their families f_1, f_2, f_3 *and durations* $d_1 = 4, d_2 = 3, d_3 = 5$*. For this problem there is a solution of the sequence dependent setup times scheduling problem with* $s_1 = 0, s_2 = 5, s_3 = 9$ *not determining a valid solution of the according multi-resource scheduling problem because there two tasks will overlap:*



4 Implementation Issues

Due to the facts that

- all solutions of a multi-resource scheduling problem determine all solutions of the according sequence dependent setup times scheduling problem
- and multi-resource scheduling is an instance of task scheduling on exclusive/single resources

it seems to be obvious in constraint-based scheduling to implement a global constraint for sequent dependent setup times scheduling on the bases of task scheduling constraints for exclusive – or often called *single* – resources. Consequently, the chosen object-oriented implementation of a class of such constraints in the object-oriented, finite-domain constraint-solving library `firstCS` [HMSW03, Wol06] uses the already existing

$$\mathsf{SingleResource}(S)$$

constraints, where S is a non-empty set of tasks to be scheduled without temporal overlapping, i.e. in linear order. In detail, within the implemented class SetupTimeOnSingleResource an according constructor

$$\mathsf{SetupTimeOnSingleResource}(A^{F\times F}, M_{T\to F}, T, B)$$

is realized where $A^{F\times F}$ is a setup time matrix, $M_{T\to F}$ is a function mapping the tasks in T to their families in F and B is set of breaks. This constructor implements the presented model for the multi-resource scheduling problem according to $(T, F, B, A^{F\times F})$ (cf. Sect. 3.4). Sum constraints are used for the *start-duration-end* and the *offset* conditions and SingleResource constraints for the *non-overlapping* condition.

The implementation of the Sum constraints is based on interval arithmetics (cf. [SS01]) pruning the finite integral domains of the variables of the sum's addends. For SingleResource constraints pruning based on *forbidden regions*, *edge finding*, *not-first/not-last detection* is applied (cf. [Vil04, Vil07, Wol03]).

In order to satisfy the *common order* condition an additional pruning method – called commonOrder – is implemented (cf. Alg. 2). It uses *detectable precedences* [Vil04, Vil07] between two activities: An activity x is (detectable) *before* another activity y if $\mathsf{ect}_y > \mathsf{lst}_x$ holds, i.e. y cannot be completed before x starts. Detectable precedence between two tasks or breaks on one of the resources of the multi-resource scheduling problem is checked by Alg. 1. There, we have to distinguish the cases whether the considered activities are task or breaks: their earliest completion time (EST) respective their latest start time (LST) are adapted accordingly. Then, precedence is checked.

Algorithm 2 prunes the domains start and end times of the tasks and breaks according to the *common-order* condition. Therefore, each pair of activities is checked for a detectable precedence using Alg. 1. Due to symmetry, only one possible order is considered (line 4). If it holds on one resource (identified by a family g) then it must hold on all other resources,

Algorithm 1: $\mathsf{isBeforeAt}(x, y, g)$ checks whether the task or break x is before another task or break y on the resource identified by the family g.

if *x is a task* **then**
  $\mathsf{ECT}_x = \mathsf{ect}_x + a_{f_x,g}$;
else if *x is a break* **then**
  $\mathsf{ECT}_x = \mathsf{ect}_x$;
if *y is a task* **then**
  $\mathsf{LST}_y = \mathsf{lst}_y + a_{f_y,g}$;
else if *y is a break* **then**
  $\mathsf{LST}_y = \mathsf{lst}_y$;
return $\mathsf{ECT}_y > \mathsf{LST}_x$;

too (iteration over all other families starts at line 5). Thus, there is an inconsistency if the reversed order holds on another resource (identified by a family f, cf. line 6). If the order is not detectable (at line 8) an update of the activities' time windows according to the order is performed because this may be not redundant.

The runtime complexity of the whole algorithm is $O(|T \cup B|^2 |T_F|)$. It is remarkable that the complexity is not quadradic in $|T_F|$ because the innermost loop (line 5) is iterated at most once for any pair of activities (cf. line 13).

5 Benchmark Examinations

Due to the fact that there are no benchmark instances of sequence dependent setup time scheduling problems with time windows and breaks available, some often examined instances without breaks are considered: the job shop scheduling instances introduced in [BT96]. These instances are modifications of the classical Lawrence instances [Law84] devoted to job shop scheduling, additionally introducing sequence dependent setup times (SDST-JSP). Each instance is characterized by a triple $m \times j \times t$ where m is the number of machines, j is the number of jobs to be scheduled on those machines and t is the number of different setup types, i.e. families. Each job consists of m tasks which have to be scheduled in linear order on all m machines. The challenge is to compute a schedule with minimal make-span, i.e. to minimize the latest completion time of all jobs $C_{\max}$ and to prove its optimality.

These problem instances are somehow task scheduling instances with time windows because the time windows of the tasks are implicitly restricted by their order within their jobs. However, these time windows depend on the bounds of the make-span.

For the minimization of the objective $C_{\max}$ a branch & bound approach with a dichotomic bounding scheme is used. Given a lower bound lwb and an upper bound upb of the objective with $\mathsf{lwb} \leq \mathsf{upb}$ an attempt is made to find a solution with $C_{\max} \leq \mathsf{mid}$ where

Algorithm 2: $\mathsf{commonOrder}()$ prunes the domains start and end times of the tasks and breaks according to the *common-order* condition.

for *each task or break* $x \in T \cup B$ **do**
  for *each task of break* $y \in (T \cup B) \setminus \{x\}$ **do**
    for *each family* $g \in F_T$ **do**
      if $\mathsf{isBeforeAt}(x, y, g)$ **then**
        for *each family* $f \in F_T \setminus \{g\}$ **do**
          if $\mathsf{isBeforeAt}(y, x, f)$ **then**
            `//` x `is before` y `at` g `but` y `is before` x `at` f
            return failure due to inconsistency;
          else if $\neg\mathsf{isBeforeAt}(x, y, f)$ **then**
            `//` x `is not yet before` y`,`
            `// thus update their time windows accordingly:`
            $\mathsf{est}_y = \max(\mathsf{est}_y, \mathsf{est}_x + d_x)$;
            $\mathsf{ect}_y = \max(\mathsf{ect}_y, \mathsf{ect}_x + d_x)$;
            $\mathsf{lst}_x = \min(\mathsf{lst}_x, \mathsf{lst}_y - d_x)$;
            $\mathsf{lct}_x = \min(\mathsf{lct}_x, \mathsf{lct}_y - d_x)$;
        break;

$\mathsf{mid} = \left\lfloor \frac{\mathsf{lwb}+\mathsf{upb}}{2} \right\rfloor$ is the mean of both bounds. If such a solution exists, the upper bound is decreased, i.e. $\mathsf{upb} = C_{\max} - 1$. If there is no solution, the lower bound is increased, i.e. $\mathsf{lwb} = \mathsf{mid} + 1$. The search continues until $\mathsf{lwb} > \mathsf{upb}$ holds. Then, the most recently found solution is a minimal solution of the considered SDST-JSP instance. The used branching is a left-to-right, depth-first incremental tree search which avoids re-traversals of already visited paths in the search tree containing suboptimal solutions (cf. [vHlP91]). The applied heuristic search strategy is specialized for job shop scheduling [Wol05]: The machine with highest *demand* is considered first. Here, the *demand* is the ratio of the sum of durations and the difference between the latest possible completion time and the earliest start time of all its tasks. Then all tasks are sorted on the current machine such that their duration is not decreasing. Then, the pairs of the first and second activity, the first and third, etc. will be ordered partially (cf. [Wol05]) until all tasks on each machine are in linear order (cf. column "sorted statically" in Table 1)

For runtime comparison only the (small) $5 \times 10 \times 5$ SDST-JSP instances t2-ps01, …, t2-ps05 and the (medium) $5 \times 15 \times 5$ instances t2-ps06, …, t2-ps10 are considered. The algorithms are coded in Java and the tests are performed on a laptop PC Intel Core 2 Duo (T7700) at 2.4 GHz with 2 GByte memory running Windows XP Professional, version 2003, SP3 and Sun Java, version 1.6.0, revision 13. The parameters and results are presented in Table 1. For branch & bound the initial lower and upper bounds (LB0 and UB0) are taken from [AF08, Table 2]. The best computed make-span ($C_{\max}$) and the

Table 1: Best computed make-span value $C_{\max}$ and elapsed runtime

Instance	LB0	UB0	Strategy 2 in [AF08]		sorted statically		GA+LS in [GVV08]	
			$C_{\max}$	CPU sec.	$C_{\max}$	CPU sec.	$C_{\max}$	CPU sec.
t2-ps01	433	844	**798***	56.7	**798***	2.1	**798***	0.7
t2-ps02	434	992	**784***	242.3	**784***	7.2	**784***	0.7
t2-ps03	359	946	**749***	699.3	**749***	15.1	**749***	0.8
t2-ps04	399	921	**730***	251.6	**730***	3.8	**730***	0.7
t2-ps05	390	733	**691***	58.2	**691***	2.7	**691***	1.0
t2-ps06	433	1120	**1009***	1797.6	**1009***	1481.8	1026	4.7
t2-ps07	416	1129	**970***	781.8	**970***	7538.2	**970***	3.3
t2-ps08	399	1066	**963***	349923.0	–	$>$ 12 h	**963***	4.3
t2-ps09	412	1174	1061	169582.0	<u>**1060***</u>	31812.1	**1060***	3.5
t2-ps10	463	1187	**1018***	35.1	**1018***	1788.9	**1018***	3.2

bold value: best known result of the instance is reached.
<u>underlined value</u>: optimality is proven the first time.
* optimal.

elapsed runtime (CPU sec.) are compared with state-of-the-art approaches presented in [AF08, GVV08]. It seems that the target machines used for the benchmark computations are comparable: In [AF08] the algorithms are coded in C++ and run on a PC with AMD64 architecture under Linx; in [GVV08] the algorithms are coded in C++ and run a PC Intel Core 2 Duo at 2.6 GHz. However, the solution approaches as well as their runtime differ. While in [AF08] constraint-based, branch & bound algorithms are used, an hybrid search combining genetic algorithms and local search (GA+LS) is applied in [GVV08]. The latter is much faster but the proofs of optimality are missing, i.e. they are impossible with such an approach. The comparison shows that the approach presented here is in general faster than that presented in [AF08] but significantly slower than the presented in [GVV08]. A reason might be that the runtime given in column "sorted statically" in Table 1 includes the required time for proving optimality which pays off if the quality of a solution is in the focus: To the best of one's knowledge, it is the first time that the optimality of the minimal make-span of problem instance t2-ps09 (namely 1060) is proven.

From a practical point of view, the significance of the benchmark comparison is questionable: The comparison is performed on artificial job shop scheduling problem instances with sequence dependent setup times. However, it is an open question how the compared approaches will perform on real scheduling problem where the time windows of the tasks are a-priori fixed and the tasks are constrained by several other conditions, too.

6 Conclusion and Future Work

A constraint-based modeling approach for task scheduling with sequence dependent setup times is extended to deal properly with work breaks. The extended model is supported by some additional pruning algorithms which perform well compared to another constraint based scheduling approach but worse than hybrid search approach combining genetic algorithms with local search. The comparison is performed on generated job shop scheduling problem instances with sequence dependent setup times. However, it is an open question how the compared approaches will perform on real scheduling problems.

Currently the presented model and algorithms are evaluated in *field workforce scheduling* performed for the maintenance of water distribution networks [SR08]. Future work focuses on a further generalization the developed constraint-based modeling, pruning and search approaches for task scheduling with sequence dependent setup times on alternative resources. Such scheduling problems are of high practical relevance, e.g. in all field workforce scheduling applications where vehicle fleets have to be managed such that the workload is distributed well over the good or even optimal vehicle routes.

References

[ABF04] Christian Artigues, Sana Belmokhtar, and Dominique Feillet. A New Exact Solution Algorithm for the Job Shop Problem with Sequence-Dependent Setup Times. In Jean-Charles Régin and Michel Rueher, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems First International Conference, CPAIOR 2004, Nice, France, April 20-22, 2004. Proceedings*, volume 3011 of *Lecture Notes in Computer Science*, pages 37–49. Springer-Verlag, 2004.

[AF08] Christian Artigues and Dominique Feillet. A branch and bound method for the job-shop problem with sequence-dependent setup times. *Annals of Operations Research*, 159(1):135–159, 2008.

[BL98] Jean-Louis Bouquard and Christophe Lenté. Equivalence to the Sequence Dependent Setup Time Problem. In Jeremy Frank and Mihaela Sabin, editors, *CP'98 Workshop on Constraint Problem Reformulation*. online available at `http://ti.arc.nasa.gov/ic/people/frank/bouquard.cp98.FINAL.ps`, October 30 1998. (Last visited: 2008/08/14).

[BT96] Peter Brucker and Olaf Thiele. A branch & bound method for the general-shop problem with sequence dependent setup-times. *OR Spectrum*, 18(3):145–161, September 1996.

[GPG01] Caroline Gagné, Wilson L. Price, and Marc Gravel. Scheduling a Single Machine with Sequence Dependent Setup Time Using Ant Colony Optimization. Technical Report 2001-003, Faculté des Sciences de L'Administration, Université Laval, Québec, Canada, April 2001.

[GVV08] Miguel A. González, Camino R. Vela, and Ramiro Varela. A New Hybrid Genetic Algorithm for the Job Shop Scheduling Problem with Setup Times. In Jussi Rintanen, Bernhard Nebel, Christopher J. Beck, and Eric A. Hansen, editors, *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling, ICAPS 2008*, pages 116–123, Sydney, Australia, September 14–18 2008. Association for the Advancement of Artificial Intelligence AAAI.

[HMSW03] Matthias Hoche, Henry Müller, Hans Schlenker, and Armin Wolf. `firstCS` - A Pure Java Constraint Programming Engine. In Michael Hanus, Petra Hofstedt, and Armin Wolf, editors, *2nd International Workshop on Multi-paradigm Constraint Programming Languages – MultiCPL'03*. Online available at `uebb.cs.tu-berlin.de/MultiCPL03/Proceedings.MultiCPL03.RCoRP03.pdf`, 29th September 2003. (Last visited: 2009/05/13).

[Law84] S. Lawrence. Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (supplement). Technical report, Graduate School of Industrial Administration, Carnegie Mellon University, 1984.

[LVA+03] D. Lesaint, C. Voudouris, N. Azarmi, I. Alletson, and B. Laithwaite. Field workforce scheduling. *BT Technology Journal*, 21(4):23–26, 2003.

[SR08] Andreas Schulz and Georg Ringwelski. A Workforce-Scheduling Application Using a CP-AI-Hybrid. In Sibylle Schwarz, editor, *22nd Workshop on (Constraint) Logic Programming*, Technical Report 2008/08, pages 83–88, Dresden, Germany, September 30 - October 1 2008. Society for Logic Programming (GLP e.V.), University Halle-Wittenberg, Institute of Computer Science.

[SS01] Christian Schulte and Peter J. Stuckey. When Do Bounds and Domain Propagation Lead to the Same Search Space. In Harald Søndergaard, editor, *Third International Conference on Principles and Practice of Declarative Programming*, pages 115–126, Florence, Italy, September 2001. ACM Press.

[VB02] Petr Vilím and Roman Barták. Filtering Algorithms for Batch Processing with Sequence Dependent Setup Times. In Malik Ghallab, Joachim Hertzberg, and Paolo Traverso, editors, *Proceedings of the 6th International Conference on AI Planning and Scheduling, AIPS'02*, pages 312–321. The AAAI Press, 2002.

[vHlP91] Pascal van Hentenryck and Thierry le Provost. Incremental Search in Constraint Logic Programming. *New Generation Computing*, 9(3 & 4):257–275, 1991.

[Vil02] Petr Vilím. Batch Processing with Sequence Dependent Setup Times: New Results. In *Proceedings of the 4th Workshop of Constraint Programming for Decision and Control, CPDC'02*, Gliwice, Poland, 2002.

[Vil04] Petr Vilím. $O(n \log n)$ Filtering Algorithms for Unary Resource Constraint. In Jean-Charles Régin and Michel Rueher, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, First International Conference, CPAIOR 2004, Proceedings*, volume 3011 of *Lecture Notes in Computer Science*, pages 335–347, Nice, France, April 20-22 2004. Springer Verlag.

[Vil07] Petr Vilím. *Global Constraints in Scheduling*. PhD thesis, Charles University in Prague, Faculty of Mathematics and Physics, Department of Theoretical Computer Science and Mathematical Logic, KTIML MFF, Universita Karlova, Malostranské náměstí 2/25, 118 00 Praha 1, Czech Republic, August 2007.

[Wol03] Armin Wolf. Pruning while Sweeping over Task Intervals. In Francesca Rossi, editor, *Principles and Practice of Constraint Programming – CP 2003, 9th International Conference*, volume 2833 of *Lecture Notes in Computer Science*, pages 739–753. Springer Verlag, 2003.

[Wol05] Armin Wolf. Better Propagation for Non-reemptive Single-Resource Constraint Problems. In B. Faltings, A. Petcu, F. Fages, and F. Rossi, editors, *Recent Advances in Constraints, Joint ERCIM/CoLogNET International Workshop on Constraint Solving and Constraint Logic Programming, CSCLP 2004, Lausanne, Switzerland, June 23-25, 2004, Revised Selected and Invited Papers*, volume 3419 of *Lecture Notes in Artificial Intelligence*, pages 201–215. Springer Verlag, 2005.

[Wol06] Armin Wolf. Object-Oriented Constraint Programming in Java Using the Library `firstCS`. In Michael Fink, Hans Tompits, and Stefan Woltran, editors, *20th Workshop on Logic Programming, Vienna, Austria, February 22–24, 2006*, volume 1843-06-02 of *INFSYS Research Report*, pages 21–32. Technische Universität Wien, 2006.