A Computational Study of Constraint Programming Approaches for Resource-Constrained Project Scheduling with Autonomous Learning Effects

Alessandro Hill[1], Jordan Ticktin[1], and Thomas W.M. Vossen[2]

[1] Industrial and Manufacturing Engineering, California Polytechnic State University, San Luis Obispo, USA {ahill29,jticktin}@calpoly.edu

[2] Leeds School of Business, University of Colorado, Boulder, USA vossen@colorado.edu

Abstract. It is well-known that experience can lead to increased efficiency, yet this is largely unaccounted for in project scheduling. We consider project scheduling problems where the duration of activities can be reduced when scheduled after certain other activities that allow for learning relevant skills. Since per-period availabilities of renewable resources are limited and precedence requirements have to be respected, the resulting optimization problems generalize the resource-constrained project scheduling problem. We introduce four constraint programming formulations that incorporate the alternative learning-based job durations via logical constraints, dynamic interval lengths, multiple job modes, and a bi-objective reformulation, respectively. To provide tight optimality gaps for larger problem instances, we further develop five lower bounding techniques based on model relaxations. We also devise a destructive lower bounding method. We perform an extensive computational study across thousands of instances based on the PSPlib to quantify the impact of project size, potential learning occurrences, and learning effects on the optimal project duration. In addition, we also compare formulation strength and quality of the obtained lower bounds using a state-of-the-art constraint programming solver.

Keywords: Resource-constrained project scheduling· autonomous learning· constraint programming.

1 Introduction

On-the-job learning occurs in most projects that involve human workforce. However, its impact is rarely taken into account during the project planning stage, and has received relatively little attention in the literature. In this paper, we explore the impact of *autonomous* learning [5], which relies on the well-established phenomenon that repeatedly performing similar jobs results in a performance increase that follows the learning curve of the executing personnel. Specifically, we consider a setting where selected pairs of similar project tasks provide a potential for learning. Executing these tasks sequentially leads to increased efficiency in that the succeeding task can be completed faster than when processing the tasks in parallel or inverse order. We assume that the tasks are performed by exogenous personnel that has access to efficient knowledge transfer. In practice, it can be difficult to identify learning relations and quantify their corresponding learning potential. Therefore, we consider the case in which each task can learn from at most one other task. This leads to a more manageable identification strategy for project managers: (I) For each task, find a task that if completed beforehand would be most beneficial if such a task exists. (II) Estimate the corresponding learning effects and quantify the potential time savings. Note, however, that we do allow multiple tasks to benefit from a common predecessor. Moreover, if training on a task helps with a potential successor, then learning is likely to happen in the converse direction in practice.

Related Work. A comprehensive survey of the resource-constrained project scheduling problem (RCPSP), model variants, and corresponding algorithmic approaches is given in [14] and [37]. Mathematical programming formulations have been studied since [32], and time-indexed RCPSP formulations [1] theoretically and computationally outperform less common event-based formulations [21]. Successful implementations incorporate cutting planes and efficient separation algorithms (e.g., [8, 42]).

Over the last decade however, Constraint programming (CP) approaches have emerged as the dominant exact solution methods for the RCPSP. The efficient incorporation of lazy clause generation [10] and the cumulative resource constraint [36], followed by several hybrid acceleration techniques (e.g., see [25]) has made CP solvers the state-of-the-art when solving small to medium sized instances. The study of different CP formulations proved CP-based approaches to also be prevailing for different RCPSP variants (e.g., [22, 35]). Recently, CP has also shown to produce near-optimal schedules at very large scale when hybridized with mathematical optimization techniques [16].

In [31], learning effects are incorporated in form of both reduced resource utilization and altered job duration by considering the discrete time/cost trade-off problem (DTCTP). The DTCTP is restricted to a single non-renewable resource which is commonly limited by worker availability or budget. Each job is allowed to be executed in one of multiple possible predefined modes. The DTCTP is a special case of the multi-mode RCPSP but neither reduces to the RCPSP nor generalizes it. The setup in [31] uses the alternative modes to represent the execution of a job consuming increased amounts of resources at augmented speed. The relation between resource consumption and duration of the different modes of a job is follows an underlying (resource) learning curve. The authors devise a genetic algorithm, compare its performance to existing approaches for the non-learning case, and conclude that learning effects notably affect the obtained project schedules. A bi-objective variant of the multi-skill RCPSP with project costs and learning benefits that stem from increased resource utilization is considered in [18].

Learning effects have been studied in scheduling problems without resource considerations, such as single-machine scheduling problems [3, 27, 33]. An overview of scheduling problems that incorporate learning is given in [2, 5]. Deterioration is a related concept that also considers dynamic job durations ([13]), while more general job start time-dependent durations are studied in [40]. For an overview of applications of learning in production and operations management we refer to [11]. In [28], a corresponding review with respect to human factors is provided.

To the best of our knowledge, there is little research that considers the impact of learning in resource-constrained scheduling. In [38], learning effects are incorporated in form of both reduced resource utilization and altered job duration by considering the discrete time/cost trade-off problem (DTCTP). The DTCTP is closely related to the RCPSP but is restricted to a single non-renewable resource which is commonly limited by a worker availability or a budget, and each job can be executed in one of multiple possible predefined modes. The DTCTP is a special case of the multi-mode RCPSP but neither reduces to nor generalizes the RCPSP. The setup in [38] uses the alternative modes to represent the execution of a job consuming increased amounts of resources at augmented speed. The relation between resource consumption and duration of the different job modes follows an underlying (resource) learning curve. For a review on related time/cost trade-off models we refer to [39]. Alternative job durations representing learning effects are related to job crashing in resource-constrained project scheduling [9], and can be interpreted as a rapid variant of plateauing for learning curves [41].

Contribution. There has been considerable research that evaluates the strength and lower bounds of different IP formulation alternatives. However, comparisons of alternative approaches to formulating CPs have received substantially less attention, and we believe this is an important aspect of our work. In addition, we carefully quantify the remarkable lower bounding capabilities of state-of-the-art CP solvers for RCPSPs which are largely underexplored. Overall, our main contributions are as follows.

- We define a novel RCPSP variant that incorporates autonomous learning;
- We propose problem reduction techniques for the resulting problem;
- We introduce four CP formulations, and provide an empirical comparison of their scheduling and lower bounding performance;
- We consider various lower bounding techniques based on model relaxations and destructive lower bounding, and empirically evaluate their strengths;
- We quantify the potential benefits of learning, and perform an extensive computational study based on a comprehensive set of PSPlib-based instances using a broad set of parameters.

The remainder of this paper is organized as follows. In Section 2, we formally define and illustrate the learning-enhanced optimization model. Different CP formulations are introduced in Section 3, and Section 4 suggests various model relaxations as well as a destructive lower bounding method. A comprehensive computational analysis for both the model and the developed approaches is conducted in Section 5, followed by our conclusion in Section 6.

2 Optimization Model

In the following, we formally define the resource-constrained project scheduling problem with learning, which we refer to as RCPSP+L. Let $T = \{0, ..., z\}$ be a discrete time horizon. We are given a set of non-preemptive jobs $J = \{1, \dots, n\}$ and each job $j \in J$ has a duration $d_j \in \mathbb{Z}_0^+$. A set of renewable resources is denoted by R and each resource $r \in R$ has a per-period availability of q_r. Each job $j \in J$ consumes $u_{j,r} \in \mathbb{Z}_0^+$ units of resource $r \in R$ in each period that it is processed ($u_{j,r} \leq q_r$). Let $A \subset J \times J$ be a set describing a precedence relation between pairs of jobs. For $(i, j) \in A$, i is called the predecessor and j the successor. We presume that the precedence digraph (J, A) is acyclic, and that job 1 (n) has duration zero, no resource utilization, and is the only job without predecessors (successors).

Furthermore, we are given a learning relation $L \subseteq J \times J$, and *learning potentials* $l_{i,j} \in \{0, \dots, d_j - 1\}$ for $(i, j) \in L$. We assume that the node in-degree in the learning digraph (J, L) is at most one. The latter will translate to the property that a job can benefit from the experience gained by the execution of at most one preceding job.

A schedule S is an assignment of start and end times in T to the jobs in J. Let $s(j, S)$ denote the start time, $e(j, S)$ the end time, and $d(j, S) = e(j, S) - s(j, S)$ the duration of $j \in J$ in S. Then a feasible schedule, or solution, for the RCPSP+L is a schedule S such that

- the total consumption of resource r in each period does not exceed the corresponding resource availability q_r for $r \in R$,
- job j does not start before job i ends for $(i, j) \in A$, and
- the duration of job j equals $d_j - l_{i,j}$ if job i precedes j, and d_j, otherwise.

The RCPSP+L asks for a feasible schedule S of minimal makespan; i.e., the latest job end time, $\max_{j \in J} e(j, S)$, is minimized. In the following, we denote the alternative duration $d_j - l_{i,j}$ of job j as d'_j and assume that $l_{i,j} = 0$ if $(i, j) \notin L$. Note that the RCPSP+L reduces to the RCPSP in the case that $L = \emptyset$, or $l_{i,j} = 0\ \forall (i, j) \in L$. Accordingly, the RCPSP+L is strongly NP-hard as the RCPSP is strongly NP-hard [6]. Moreover, the relation L is neither transitive nor symmetric. We define the *learning frequency* ϕ for a problem instance as the relative number of jobs with a learning potential; i.e., $\phi = |L|/|J|$. The *learning intensity* λ is defined as the average of the learning potentials; i.e., $\lambda = \sum_{(i,j) \in L} l_{i,j}/|L|$. Let $i \prec j$ denote that there exists a directed path from i to j in (J, A) for $i \neq j \in J$.

Example To illustrate the impact of learning, consider an instance of the RCPSP+L with $J = \{1, \dots, 8\}$ and $R = \{1, 2\}$. Let the precedence digraph with job durations, the learning digraph with learning potentials, and the resource requirements and availabilities be as depicted in Figure 1. An optimal schedule for the corresponding RCPSP instance (i.e., when $L = \emptyset$) with makespan 12 is illustrated in Figure 2 (left). However, the integration of learning in the RCPSP+L allows an optimal schedule S' with makespan 9 (Figure 2, right). It can be seen that job 5 and job 6 benefit from shortened durations ($d(5, S') = d(6, S') = 2$) since they are scheduled after job 4 and job 7, respectively ($l_{4,5} = 2$, $l_{7,6} = 3$). Note that the duration of job 3 remains unchanged since job 7 is not finished before job 3 starts.

To conclude this section, we describe two techniques that focus on preprocessing of learning relations in an RCPSP+L instance.

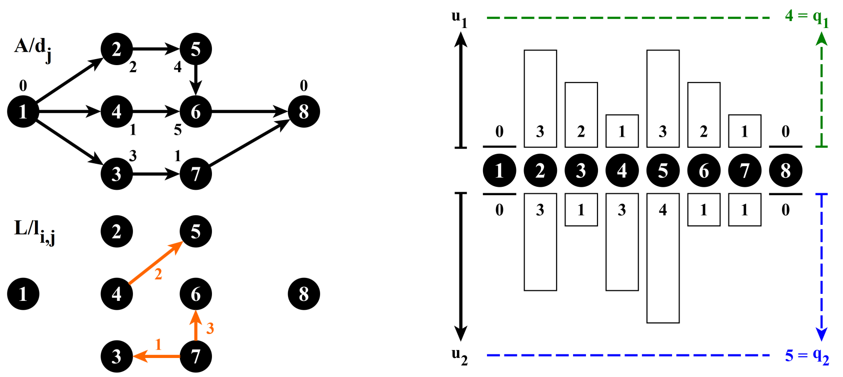

Fig. 1. Precedences with job durations (left, upper), learning digraph with learning potentials (left, lower), and resource details (right), for an example RCPSP+L instance.

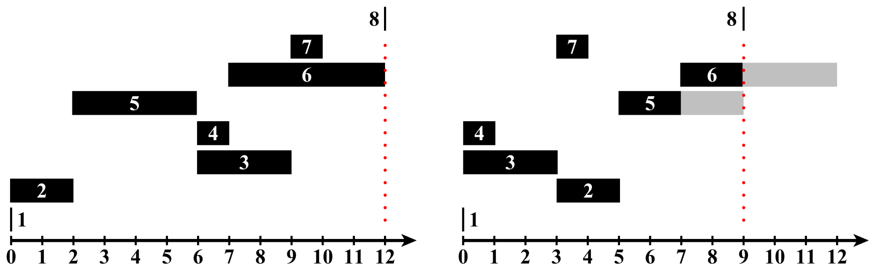

Fig. 2. An optimal solution with makespan 12 for an RCPSP instance (left) and an optimal solution with makespan 9 for the learning-enhanced RCPSP+L instance (right).

Precedence-Induced Learning Effects. A learning effect might be implied by the presence of a corresponding (in-)direct precedence requirement. In this case the following implication holds for every feasible schedule S.

$$\big((i,j) \in L \wedge i \prec j\big) \implies \big(d(j,S) = d_j - l_{i,j}\big) \qquad \forall i \neq j \in J$$

Consequently, we can set $d_j := d'_j$ and $L := L \setminus \{(i,j)\}$. The test can be performed efficiently via depth-first-search in (J, A) for each (i,j) in $\mathcal{O}(|L|(V+E))$.

Directed Cycle Elimination. In both the RCPSP and the RCPSP+L, it is assumed that A is acyclic since circular precedences cannot be implemented in any feasible schedule. Therefore, an arc $(i,j) \in L$ can be removed in an instance of the RCPSP+L if $(J, A \cup \{(i,j)\})$ contains a directed cycle. To efficiently check for these directed cycles, it is sufficient to check whether j is a direct or indirect predecessor of i in A using depth first search. Formally, we obtain:

$$\big((i,j) \in L \wedge j \prec i\big) \implies \big(d(j,S) = d_j\big) \qquad \forall i \neq j \in J.$$

3 Constraint Programming Formulations

In this section, we propose four CP formulations for the RCPSP+L that extend an efficient formulation for the RCPSP. Our objective is twofold. First, we demonstrate that using existing constraints with their propagators can lead to notable resolution performance variation. Second, we empirically identify an efficient formulation that can be implemented with minimal effort. Even though we use the term formulation, it is important to emphasize that we are considering constraint programs. As pointed out in [29], the latter generally describes a computer program rather than a mathematical description of the problem as it is done in integer programming (IP). For an in-depth comparison of CP and IP, we refer to [15, 29, 34].
We consider a well-known and efficient CP formulation [23, 25] for the RCPSP that uses an interval variable y_j to represent each job $j \in J$. We access its start time, end time, and duration by $\texttt{start}(y_j)$, $\texttt{end}(y_j)$, and $\texttt{length}(y_j)$.

$$(F_0)\ Min \quad \max_{\texttt{j}\in\texttt{J}}\big(\texttt{end}(y_j)\big) \tag{1}$$

$$subject\ to \quad \texttt{cumulative\_function}\big((y_1, q_{1,r}), \dots, (y_n, q_{n,r})\big) \ \leq\ q_r \ \forall r \in R, \tag{2}$$

$$\texttt{end\_before\_start}(y_i, y_j) \qquad \forall (i,j) \in A, \tag{3}$$

$$y_j \ \texttt{interval variable in}\ [0, z]\ \texttt{of length}\ d_j \qquad \forall j \in J. \tag{4}$$

The objective (1) minimizes the latest job end time. Inequalities (2) ensure that the per-period capacity is not exceeded for any resource using a cumulative function which represents the total resource usage by all jobs in each period. The efficient propagation of resource consumption is a key for solver scheduling performance [36]. The end-start precedence relation given in A is enforced by the precedence constraints (3). We assume that the interval variables introduced in (4) have start and end times in T and fixed durations.

Logical Formulation We first present a logic-based incorporation of learning into formulation (F_0). Then the RCPSP+L can be formulated as follows.

$$(F_1)\ Min \quad \max_{\texttt{j}\in\texttt{J}}\big(\texttt{end}(y_j)\big) \tag{5}$$

$$subject\ to \quad (2),\ (3), \tag{6}$$

$$\big(\texttt{end}(y_i) \leq \texttt{start}(y_j)\big) \implies \big(\texttt{length}(y_j) = d'_j\big) \qquad \forall (i,j) \in L, \tag{7}$$

$$\big(\texttt{end}(y_i) > \texttt{start}(y_j)\big) \implies \big(\texttt{length}(y_j) = d_j\big) \qquad \forall (i,j) \in L, \tag{8}$$

$$y_j \ \texttt{interval variable in}\ [0, z]\ \texttt{of length}\ [d'_j, d_j] \qquad \forall j \in J. \tag{9}$$

Implications in constraints (7) and (8) enforce that the alternative duration is used if the precedence requirement is met; otherwise, the original duration is imposed. The duration of interval variables introduced in (9) is between the job's alternative duration and the original duration. The following RCPSP+L formulations reduce to formulation (F_0) if $L = \emptyset$.

Dynamic Duration Formulation Instead of using the logical constraints (7) and (8) to implement learning effects, we can dynamically subtract the learning-based reduction from the job durations. This yields the following formulation.

$$(F_2)\ Min \quad \max_{\mathtt{j}\in \mathtt{J}}\big(\mathtt{end}(y_j)\big) \tag{10}$$

$$subject\ to \quad (2),\ (3),\ (9), \tag{11}$$

$$\mathtt{length}(y_j) \ = \ d_j - l_{i,j} * \big[\mathtt{end}(y_i) \leq \mathtt{start}(y_j)\big] \qquad \forall (i,j) \in L. \tag{12}$$

The length of an interval variable (equation (12)) is set to the original job duration minus the learning potential if a learning precedence requirement is met.

Multi-Mode Formulation The two different durations in case of a learning potential can be interpreted as two modes in which the job can be processed. The multi-mode resource-constrained project scheduling problem (MM-RCPSP) is a well-known generalization of the RCPSP that allows such multiple modes. In addition to the broad applicability of the MM-RCPSP itself, it was shown that MM-RCPSP reformulations of optimization problems yield computationally efficient approaches (e.g, [17]). We formulate the RCPSP+L using multiple modes by introducing two optional interval variables z_j and z'_j for each job j to represent the two modes.

$$(F_3)\ Min \quad \max_{\mathtt{j}\in \mathtt{J}}\big(\mathtt{end}(y_j)\big) \tag{13}$$

$$subject\ to \quad (2),\ (3),\ (9), \tag{14}$$

$$\mathtt{alternative}(y_j, [z_i, z_j]) \qquad \forall (i,j) \in L, \tag{15}$$

$$\mathtt{presence\_of}(z'_j) \ = \ \big(\mathtt{end}(y_i) \leq \mathtt{start}(y_j)\big) \qquad \forall (i,j) \in L, \tag{16}$$

$$z_j, z'_j \ \mathtt{optional} \ \ \mathtt{interval\ vars\ in}\ [0,z]\ \mathtt{of\ length}\ d_j, d'_j \ \ \forall (i,j) \in L. \tag{17}$$

The alternative constraint (15) effects that exactly one of the two variables z_j and z'_j is present, and that its length equals the length of y_j. The left-hand side of equation (16) consists of a `presence_of` expression which takes value 1 if variable z'_j is present in a solution and 0, otherwise. The latter is equated with the logical precedence expression used in the right-hand side of equation (12).

Bi-Objective Reformulation Motivated by providing additional guidance to the CP solver, we reformulate the RCPSP+L as a bi-objective optimization problem that also maximizes the learning utilization. We apply an a priori method that ranks the makespan objective over the secondary objective by modifying formulation (F_2) as follows.

$$(F_4)\ Min \quad \max_{\mathtt{j}\in \mathtt{J}}\big(\mathtt{end}(y_j)\big) \ - \ \frac{l}{|J|+1} \tag{18}$$

$$subject\ to \quad (2),\ (3),\ (9),\ (12), \tag{19}$$

$$\mathtt{int}\ l = \sum_{j\in J} \big[\mathtt{length}(y_j) < d_j\big]. \tag{20}$$

The integer variable l (Eq. (20)) measures the number of active learning effects. In objective function (18), the minimization of the project makespan dominates the maximization of l. However, incrementing the number of learning effects ameliorates the objective function by $1/(|J|+1)$.

4 Relaxations, Restrictions and Lower Bounding

In this section, we introduce various relaxations for the RCPSP+L, and describe a destructive lower bounding method. We also suggest an approach to obtain upper bounds based on imposing a problem restriction. For an overview of lower bounding techniques for the RCPSP and their use, we refer to [7, 30, 37]. Relaxations strengthened by Lagrangian dualization for the related net present value RCPSP are studied in [12]. The subsequent bounding techniques are implemented as standalone methods and their individual results are analyzed in Section 5.

4.1 CP-Based Lower Bounding

State-of-the-art CP solvers store and dynamically update lower bounds in order to prune effectively. Besides lower bounds that are raised during the core branch-and-bound method and through inference, strong bounds are derived via the efficient solution of relaxations (e.g., [24]). Recent developments have led to significant, albeit CP solver and model-dependent, performance improvements. Currently, this strength appears to be largely unknown. We will provide quantitative evidence for the quality of this bound, and use these bounds as an optimality guarantee for unsolved instances.

4.2 Relaxations

Let $opt(P)$ be the optimal objective function value for problem instance P, and $lb(P)$ ($ub(P)$) describe a valid lower (upper) bound for $opt(P)$. Let us begin with a note on the minimal project makespan for an RCPSP+L instance P. Certainly, every job that is shortened due to a learning effect is preceded by an activity that facilitates the learning. Therefore, at least one job has to be executed with its original duration; or, in other words, not all jobs in a schedule can benefit from learning. Assuming that $d_j > 0$, this yields a lower bound: $opt(P) \geq \min_{j \in J} d_j$.

An RCPSP Relaxation. Let RCPSP$^-$(P) denote the RCPSP obtained from an RCPSP+L instance P after dropping the learning effects and setting the static job durations to be the reduced ones (i.e., $L := \emptyset$ and $d_j := d'_j \ \forall (i,j) \in L$). Since the optimal makespan for P never exceeds the optimal makespan for RCPSP$^-$(P), every lower bound for the RCPSP$^-$ is a valid lower bound for RCPSP+L: For an RCPSP+L instance P, it holds that $lb(RCPSP^-(P)) \leq opt(P)$. Consequently, every lower bounding technique for the RCPSP can be applied to derive a lower bound for the RCPSP+L.

A Project Scheduling Relaxation. A classical relaxation for the RCPSP is obtained by the removal of the resources, leading to the well-known project scheduling problem (PSP). In our setting, the obtained lower bound is only valid for an RCPSP+L instance P, if derived from RCPSP$^-$(P). We denote the corresponding PSP obtained for P by PSP$^-$(P). For an RCPSP+L instance P, it holds that $opt(PSP^-(P)) \leq opt(P)$. It is well known that the PSP can be solved efficiently via topological sorting [26] of the acyclic digraph (J, A).

A Learning Project Scheduling Relaxation. Let PSP+L(P) define the project scheduling problem with learning effects derived from an RCPSP+L instance P that does not consider resources; i.e., $R = \emptyset$. Note that this model incorporates our learning concepts into the classical project scheduling problem. For an RCPSP+L instance P, it holds that $lb(PSP + L(P)) \leq opt(P)$.

A Resource-Constrained Scheduling Problem Relaxation. We consider the scheduling problem with learning effects RCSP+L(P) obtained from RCPSP+L instance P after dropping the precedences (i.e., $A := \emptyset$). This relaxation is related to multi-dimensional packing problems when considering every resource in P and the project duration as spatial dimensions. For an RCPSP+L instance P, it holds that $lb(RCSP + L(P)) \leq opt(P)$. The RCSP+L is NP-hard since the RCSP (without learning) generalizes the two-dimensional strip-packing problem which is known to be strongly NP-hard [4].

4.3 Destructive Lower Bounding

Destructive lower bounding (DLB) [19] is known to produce strong lower bounds on the optimal makespan for the RCPSP. DLB is an iterative procedure. For a given valid lower bound lb, the basic idea is to prove that no schedule with makespan lb exists. If this can be done efficiently, we can increment the current lower bound ($lb := lb + 1$), and argue that the initial value for lb was destructed. Algorithm 1 describes the method. Procedure $infeasible(P, MS \leq lb, t_{max})$ re-

Algorithm 1: Destructive Lower Bounding for the RCPSP+L

```
  Input: $P$ (RCPSP+L instance), $t_{max}$ (destruction time limit)
  Output: $lb$ (valid lower bound for $P$)
1 $lb \leftarrow 0$;
2 while $(infeasible(P, MS \leq lb, t_{max}) = true)$ do
3     $lb \leftarrow lb + 1$;
4 return $lb$;
```

turns true if infeasibility can be proven within t_{max} seconds for the RCPSP+L instance P with the additional side constraint that the makespan must not exceed lb; false otherwise. In line 1, lb can be initialized by any valid lower bound on the makespan. In practice, an efficiently computable bound, such as from Section 4.2, can be used to accelerate the method. We use CP formulation (F_2) to detect infeasibility in line 2. Note that Algorithm 1 may be modified to return an optimal schedule when replacing the latter procedure by a black-box solver that is capable of finding an optimal solution. If such an optimal solution is found, then the current *lb*-value is returned and equals the optimal objective function value.

4.4 A Restriction-Based Upper Bound

Let RCPSP$^+$(P) be the RCPSP derived from an RCPSP+L instance P by dropping the learning relations. Then every schedule S that is feasible for RCPSP$^+$(P) yields a valid upper bound for P, since allowing learning would lead to a makespan that is as least as good as the one of S. For an RCPSP+L instance P, it holds that $opt(P) \leq ub(RCPSP^+(P))$. The obtained optimization problem is NP-hard. However, we may use efficient methods that have been developed for the RCPSP. Furthermore, the described upper bounding method can be used to obtain a feasible schedule for P when computed in a constructive fashion. More detailed, every solution that is found for RCPSP$^+$(P) can be transformed to a solution for P by applying learning effects that may occur. This could be followed by a compacting step which aims at further reducing the current makespan.

5 Computational Study

We present the results of an extensive experimental study to better understand the possible impact of learning and the effectiveness of the scheduling and lower-bounding methods.

Since the RCPSP+L is a new problem, no real-world instances are available literature. We derive RCPSP+L test instances from 480 RCPSP instances with 30 jobs and 600 RCPSP instances with 120 jobs from the well-known PSPlib[20]. For each RCPSP base instance, we randomly choose jobs in an iterative fashion from J for frequency $\phi \in \{0, 0.1, \ldots, 1\}$ ($\phi \in \{0.25, 0.5, 0.75\}$ for $|J| = 120$) and intensity $\lambda \in \{0.1, 0.3, \ldots, 0.9\}$ ($\lambda \in \{0.1, 0.5, 0.9\}$ for $|J| = 120$) until $|L| = \lceil \phi |J| \rceil$. For such a learning successor job j, we randomly draw a learning predecessor i such that i is no (indirect) predecessor of j in A. The learning potential is set to $l_{i,j} = \min\{\lceil \lambda d_j \rfloor, \max\{d_j - 1, 1\}\}$ and the arc (i, j) is added to L if $l_{i,j} > 0$. We do this for three different random seeds, resulting in a set of 79,200 RCPSP+L instances (L_{30}) for 30-job RCPSP instances, and a set of 21,600 RCPSP+L instances (L_{120}) for 120-job RCPSP instances (100,800 instances in total). We include the non-learning case (RCPSP) in our study for 30-job instances ($\phi = 0$). Moreover, the target number of arcs in L may not be met in which case we stop the procedure but include the instance. This instance generation scheme ensures that no redundancies with respect to the reduction techniques occur. Computations are conducted on an Intel i5-2320, 3 GHz, 8 GB machine. We use IBM ILOG CP Optimizer 12.9.0 as CP solver, using 4 workers.

5.1 CP Formulation Comparison

We first compare formulations (F_1), (F_2), (F_3) and (F_4) computationally with respect to test set L_{120}. We limit the per-instance computation time to 60 seconds.

Figure 3 shows the percentage of instances that could be solved to optimality by each formulation (left), the average solve time (center) and the optimality gaps obtained for the remaining unsolved instances (right). Formulation (F_2)

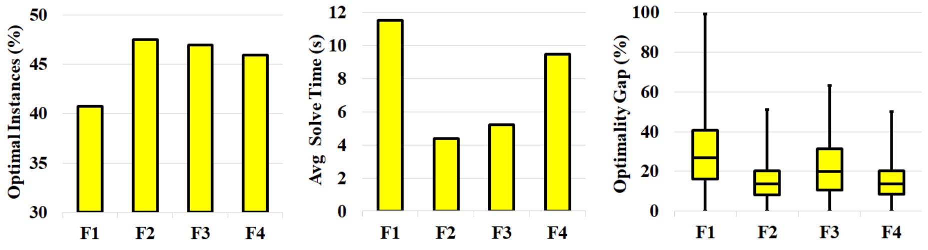

Fig. 3. The relative number of RCPSP+L instances with 120 jobs solved by each CP formulation in 60 seconds (left), the average run time for the optimally solved instances (center), and the corresponding optimality gaps for the unsolved instances (right).

solves the largest number of test instances to optimality (47.6%). Formulations (F_3) and (F_4) perform almost as well (47.0%, 46.0%) whereas formulation (F_1) produces notably less optimal schedules (40.7%). Regarding the average running time to solve instances to optimality, formulation (F_2) outperforms formulation (F_3) (4.4s<5.3s), formulation (F_4) (9.5s), and formulation (F_1) (11.5s). For 316 instances (1.5%), formulation (F_1) could not find a feasible schedule at all, which never happened for the other formulations. However, formulation (F_1) found optimal schedules for 60 instances that could not be solved by the others. Although formulation (F_2) and formulation (F_3) seem comparable, the superior performance of formulation (F_2) is further observed when considering the achieved optimality gaps for the unsolved instances in Figure 3 (right). The average optimality gap is 15.2% compared to 22.0%. Similarly, formulation (F_2) outperforms (F_4) in terms of average optimal solution times (4.4s<9.5s), as depicted in Figure 3 (center). As mentioned in Section 3, constraint programs differ from integer programs and they are typically not compared with respect to their theoretical strength. A variety of solver-dependent reduction (preprocessing), extraction (internal model representation), inference (constraint propagation) and branching strategies (no-good learning) are responsible for how efficiently they can be solved. Nevertheless, we provide some empirical insights into the solver's internal behavior. In Table 1, we show the average relative differences (Δ%) of formulation/engine variables, formulation/engine constraints, solutions found, and optimality gap (w.r.t. the best upper bound) derived from the initial lower bound for between formulation (F_2) and (F_1)/(F_3)/(F_4). Note that the engine variables (constraints) are the variables (constraints) that are internally used by the solver after extracting the formulation. For example, formulation (F_3) has three times more formulation variables than (F_2), whereas formulations (F_1) and (F_4) use identical variables sets. The solver uses a reduced number (-15.7%) of engine variables for (F_1), but more than twice as many variables are used in the other formulations. Note that (F_4) produces a larger number of solutions within

Table 1. Comparison of relative (engine) formulation size, initial #solutions, and optimality gaps from lower bounds w.r.t. formulation (F_2) for 120-job instances.

Formulation	Variables (Δ%)		Constraints (Δ%)		#Sols(Δ%)	Gap(Δ%)
	Formulation	Engine	Formulation	Engine		
F_1	0.0	-15.7	20.8	49.7	-0.4	217.6
F_3	200	115.0	35.5	-22.1	-8.0	286.1
F_4	0.0	144.6	0.0	289.5	215.2	0.0

the time limit because schedule perturbations that do not affect the makespan may still lead to alternative solutions of different objective function value due to a change in the secondary objective value. The most significant difference between formulation (F_2) and the inferior formulations (F_1) and (F_3) is the quality of the initial lower bound on the makespan. The former leads to an optimality gap that is more than three times smaller than for the others.

5.2 Lower Bounding Performance

In order to better understand the efficacy of our algorithmic approaches, we apply both lower and upper bounding techniques from Section 4 to the large instances in L_{120}. First, we illustrate how effectively the bounding models can be solved using CP in Figure 4. We use a time limit of 60 seconds per instance and model. It can be seen that problems PSP and PSP+L can be solved to optimality for all instances. Models RCPSP$^-$ and RCPSP$^+$ can be solved in 52.3% and 45.3%, respectively, of the cases, which is close to the RCPSP+L optimality rate (47.9%) obtained by formulation (F_2). This indicates that our best RCPSP+L formulations find optimal solutions efficiently. Note that formulations (F_1) and (F_4) solve a slightly lower number of instances to optimality (40.2% and 44.6%). Our CP approach has notable difficulties with the RCSP+L model for which only 10.6% of the instances can be solved. The average optimality gap for the latter model is 7.0%. We also analyze the strength of the obtained lower

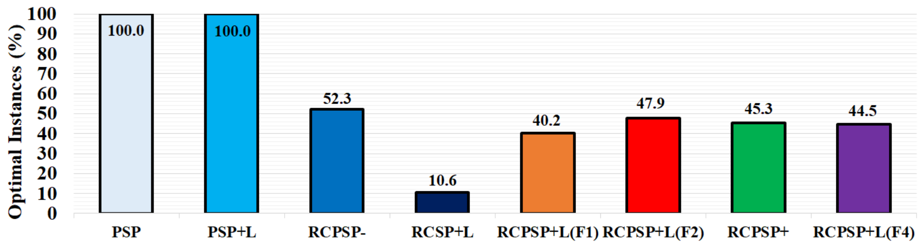

Fig. 4. The relative number of 120-job instances for which the different RCPSP+L relaxations, and formulations (F_1), (F_2) and (F_4), can be solved to optimality by CP.

bounds for the 13118 (60.7%) instances that could not be solved to optimality by formulation (F_1). Figure 5 depicts the achieved optimality gap distributions relative to the best upper bound. The time limit per bound destruction in the

DLB method is 60 seconds. The best non-DLB lower bound is always achieved

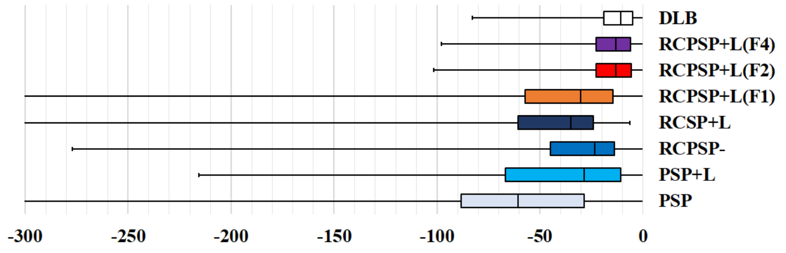

Fig. 5. The distribution of the optimality gaps achieved by the different lower bounding methods with respect to the best upper bounds for unsolved instances in L_{120}.

by a RCPSP+L CP formulation. However, in 9338 cases (71%), DLB produces an even stronger bound. For (28%) of the instances, it returns the same bound, and for only 114 instances (<1%) the computed lower bound is inferior. We note that, presumably for even larger instances, model PSP+L can be used to compute initial lower bounds very efficiently. Moreover, the makespan computed by RCPSP+L can be improved by RCPSP$^+$ for nine instances.

5.3 Scheduling and Upper Bounding Efficacy

We compute upper bounds using CP for RCPSP$^+$, formulation (F_1), formulation (F_2), and formulation (F_4). Figure 6 depicts the achieved optimality gap distributions relative to the best lower bound. Formulation (F_4) performs simi-

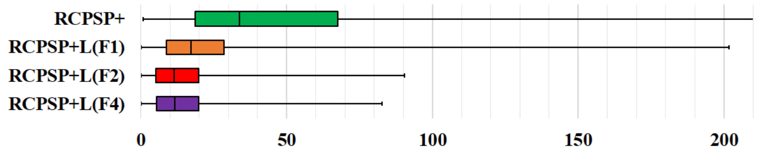

Fig. 6. The distribution of the optimality gaps achieved by the different upper bounding methods with respect to the best lower bounds for unsolved instances in L_{120}.

lar to formulation (F_2) in terms of average gap (both 13.8%). However, neither one strictly dominates the other. Formulation (F_2) produces a strictly better schedule than formulation (F_4) for 35.3% of the unsolved instances, and 28.4% vice versa. Based on these observations, we suggest using formulation (F_2) since objective function values directly represent the schedules' makespans.

5.4 Overall Performance

We summarize the achievements of our CP-based approaches in comparison to the initial formulation (F_1) in Figure 7. It shows the distribution of the optimality gaps for formulation (F_1) versus the best gaps (left), for the 120-job instances that cannot be solved by formulation (F_1).Our intention is to highlight the benefits of investigating alternative formulations, beyond formulation (F_1). We observe an 86.4% average gap reduction (96.6%$\rightarrow$13.1%). In some cases the lower bound from formulation (F_1) is extremely poor (363 times ≤ 5), causing a high average gap. We also analyzed the optimality gap reduction for all initially open 120-job instances after applying our lower and upper bounding techniques (right). The average gap reduction is 86.2% (58.1%$\rightarrow$8.0%). We tighten gaps for almost all the open instances (99.1%). Finally, we are able to optimally solve 8.3% of initially unsolved instances. For comparison, about 49% of the optimal solutions for 120-job RCPSP base instances are known. The average optimality gap for open instances is 5.7%. The 30-job RCPSP instances are known to be

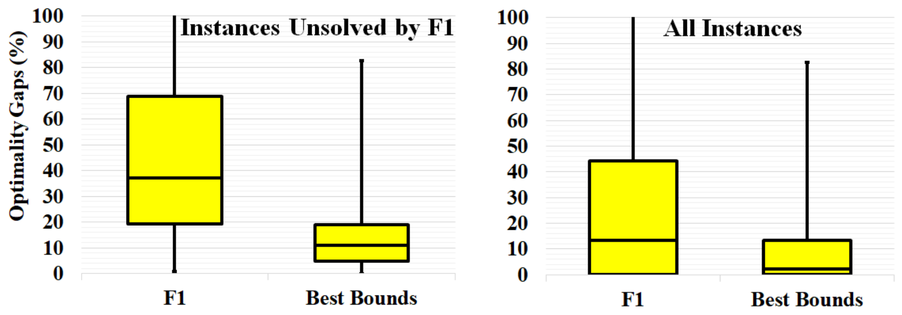

Fig. 7. The distribution of optimality gaps obtained by formulation (F_1) versus best optimality gaps from our lower bounding methods and formulations for 120-job instances; Left: instances not solved to optimality by (F_1); Right: All instances.

all solved. For 120-job projects, we improve the lower bound for 13 instances (see Appendix A). The optimality gap is reduced by 30% on average. When increasing the time limit to 600 seconds we could improve another lower bound (instance j1208_6). Note that the focus of this work is on the RCPSP+L and comprehensive computational scenario studies; but in related studies on smaller instance sets, a significantly higher time limit leads to improved results when using CP (see, for instance, [35]).

5.5 Learning Potential and Benefit

We perform an optimization-based computational model analysis using the cases in L_{30}. Our goal is to obtain insight into how pre-solving parameterization relates to actual makespan benefits. As in [31], we consider a very-large instance set to carefully capture the model behavior. All instances in L_{30} can be solved to optimality by formulation (F_2) in under 600 seconds. The average solve time is 0.85 seconds, and for 71094 (89.8%) instances, optimality can be proven within under 1 second. Therefore, we are able to accurately examine the impact of the model parameters on actual learning effects and best possible makespan for these instances. Figure 8 (left) shows the average relative makespan reductions for different values of ϕ and λ. It can be seen that the maximally achievable average makespan reduction (by 49%) is obtained at a learning frequency of $\phi = 1$ and a learning intensity $\lambda = 0.9$. When reducing the intensity to $\lambda = 0.1$ the optimal makespan is reduced by only 7% on average. For increasing ϕ, we observe that the average makespan reduction can be approximated by a linear function. Note that

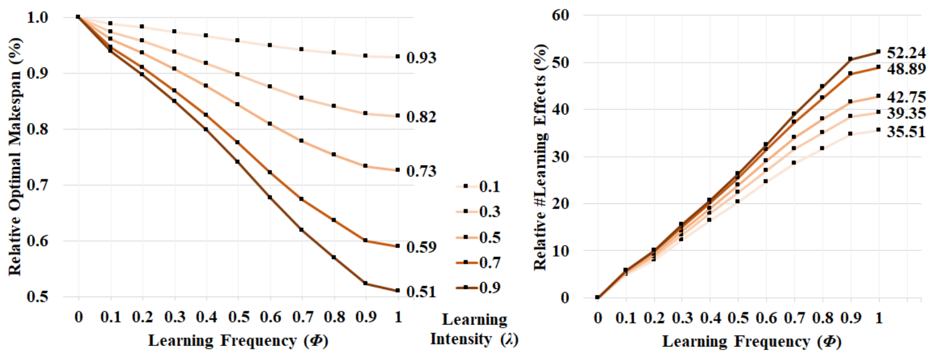

Fig. 8. Average relative optimal makespan (left), and relative #jobs that benefit from learning (right) w.r.t. instance learning frequency and intensity (30-job instances).

when $\phi \geq 0.8$, then the number of instances differs from other scenarios because the target number of arcs in L cannot not always be generated while asserting instance feasibility. To better comprehend the utilization of learning potentials in optimal schedules, the relative numbers of jobs that are actually performed in reduced time (with respect to all the jobs with learning potential) are shown in Figure 8 (right). Again, we observe an almost linear growth of actual learning effects when augmenting ϕ. The average learning utilization ranges from 35.5% to 52.2% for the different values of λ. However, the maximal utilization over all the individual instances is 83.3%. In the scenario $\phi = 1$ and $\lambda = 0.9$ the lowest value is 23.3%. In both charts we observe an irregular jump between $\lambda = 0.5$ and $\lambda = 0.7$.

5.6 Parameter Performance Impact

In the following, we analyze the impact of learning frequency ϕ and learning intensity λ on the efficacy of our techniques. We use the best upper and lower bounds obtained by all methods. Figure 9 (left) shows the percentage of instances that we solve to optimality with respect to the various learning parameter combinations. It can be seen that more instances can be solved when considering $\lambda = 0.9$. The contrast to the $\lambda \in \{0.1, 0.5\}$ cases increases when augmenting the learning frequency ϕ. As illustrated in Figure 9 (right), we achieve the best opti-

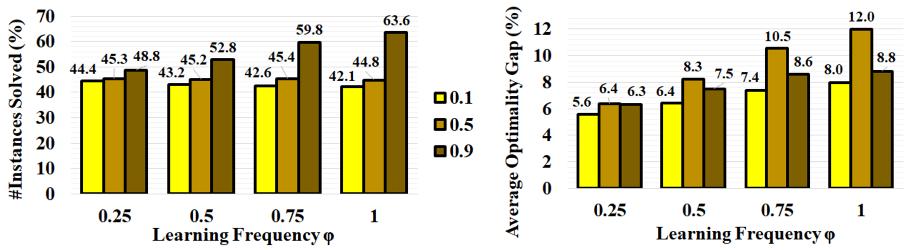

Fig. 9. The relative number of solved instances in L_{120} for the different learning parameters (left), and the corresponding average optimality gaps (right).

mality gaps for instances with a low learning frequency ($\phi = 0.25$). The hardest instances for our approaches are the ones with $\lambda = 0.5$ and $\phi = 1$. In this case (1,800 instances), the average optimality gap is 12.0%, and even reaches 82.7% in the worst case.

6 Conclusion

We introduced and studied a novel variant of the resource-constrained project scheduling problem that incorporates autonomous learning capabilities. We presented reduction techniques and four constraint programming formulations. Various lower bounding techniques were developed that require the resolution of model relaxations, as well as a destructive lower bounding approach. After conducting computational tests on more than 100,000 literature-derived test instances, we identified the most efficient formulation using a state-of-the-art constraint programming solver. Hence, we were able to optimally solve all 30-job instances, most of them in under one minute. Furthermore, we solved about half of the 120-job instances to optimality, leaving an average optimality gap of 13.1% for instances that could not be solved to optimality. We empirically analyzed the efficiency and effectiveness of the individual lower bounding methods for the unsolved problems.

Our study shows that projects can dramatically benefit from considering learning opportunities. Significant makespan reductions (up to 50%) can be achieved with ample opportunities for learning. The parameter-makespan dependency can be described as near-linear. In sum, we observe that the integration of learning potentials into resource-constrained scheduling leads to problems that can be solved by CP - when properly formulated - as efficiently as the RCPSP itself.

The resulting model represents a first step towards a new direction of research in project scheduling. The analysis of alternative or extended learning concepts, such as for example multi-predecessor learning, could be of interest. Moreover, the development of IP-based approaches could help to better understand the challenges and opportunities of integrated learning benefits in project scheduling.

References

1. Artigues, C.: On the strength of time-indexed formulations for the resource-constrained project scheduling problem. Operations Research Letters **45**(2), 154–159 (2017)
2. Azzouz, A., Ennigrou, M., Ben Said, L.: Scheduling problems under learning effects: classification and cartography. International Journal of Production Research **56**(4), 1642–1661 (2018)
3. Bai, D., Tang, M., Zhang, Z.H., Santibanez-Gonzalez, E.D.: Flow shop learning effect scheduling problem with release dates. Omega **78**, 21–38 (2018)
4. Baker, B.S., Coffman, Jr, E.G., Rivest, R.L.: Orthogonal packings in two dimensions. SIAM Journal on computing **9**(4), 846–855 (1980)

5. Biskup, D.: A state-of-the-art review on scheduling with learning effects. European Journal of Operational Research **188**(2), 315–329 (2008)
6. Blazewicz, J., Lenstra, J., Kan, A.: Scheduling subject to resource constraints: classification and complexity. Discrete Applied Mathematics **5**(1), 11–24 (1983)
7. Brucker, P., Knust, S.: Lower bounds for resource-constrained project scheduling problems. European Journal of Operational Research **149**(2), 302–313 (2003)
8. Demassey, S., Artigues, C., Michelon, P.: Constraint-propagation-based cutting planes: An application to the resource-constrained project scheduling problem. INFORMS Journal on computing **17**(1), 52–65 (2005)
9. Dodin, B., Elimam, A.: Integrated project scheduling and material planning with variable activity duration and rewards. IIE Transactions **33**(11), 1005–1018 (2001)
10. Feydy, T., Stuckey, P.J.: Lazy clause generation reengineered. In: Gent, I.P. (ed.) Principles and Practice of Constraint Programming - CP 2009. pp. 352–366. Springer (2009)
11. Glock, C.H., Grosse, E.H., Jaber, M.Y., Smunt, T.L.: Applications of learning curves in production and operations management: A systematic literature review. Computers & Industrial Engineering **131**, 422–441 (2019)
12. Gu, H., Stuckey, P.J., Wallace, M.G.: Maximising the net present value of large resource-constrained projects. In: Milano, M. (ed.) Principles and Practice of Constraint Programming. pp. 767–781. Springer (2012)
13. Gupta, J.N., Gupta, S.K.: Single facility scheduling with nonlinear processing times. Computers & Industrial Engineering **14**(4), 387–393 (1988)
14. Hartmann, S., Briskorn, D.: A survey of variants and extensions of the resource-constrained project scheduling problem. European Journal of Operational Research **207**(1), 1–14 (2010)
15. Heipcke, S.: Comparing constraint programming and mathematical programming approaches to discrete optimisation-the change problem. Journal of the Operational Research Society **50**(6), 581–595 (1999)
16. Hill, A., Brickey, A., Newman, A., Goycoolea, M.: Hybrid optimization strategies for resource constrained project scheduling problems in underground mining (2019), manuscript
17. Hill, A., Lalla-Ruiz, E., Voß, S., Goycoolea, M.: A multi-mode resource-constrained project scheduling reformulation for the waterway ship scheduling problem. Journal of Scheduling **22**(2), 173–182 (2019)
18. Hosseinian, A.H., Baradaran, V., Bashiri, M.: Modeling of the time-dependent multi-skilled RCPSP considering learning effect. Journal of Modelling in Management **14**(2), 521–558 (2019)
19. Klein, R., Scholl, A.: Computing lower bounds by destructive improvement: An application to resource-constrained project scheduling. European Journal of Operational Research **112**(2), 322–346 (1999)
20. Kolisch, R., Sprecher, A.: PSPLIB-A project scheduling problem library: OR software-ORSEP operations research software exchange program. European Journal of Operational Research **96**(1), 205–216 (1997)
21. Koné, O., Artigues, C., Lopez, P., Mongeau, M.: Event-based MILP models for resource-constrained project scheduling problems. Computers & Operations Research **38**(1), 3–13 (2011)
22. Kreter, S., Schutt, A., Stuckey, P.J.: Using constraint programming for solving RCPSP/max-cal. Constraints **22**(3), 432–462 (2017)
23. Laborie, P.: IBM ILOG CP Optimizer for detailed scheduling illustrated on three problems. In: International Conference on AI and OR Techniques in Constraint

Programming for Combinatorial Optimization Problems. pp. 148–162. Springer (2009)
24. Laborie, P., Rogerie, J.: Temporal linear relaxation in IBM ILOG CP Optimizer. Journal of Scheduling **19**(4), 391–400 (2016)
25. Laborie, P., Rogerie, J., Shaw, P., Vilím, P.: IBM ILOG CP Optimizer for scheduling. Constraints - An International Journal **23**(2), 210–250 (2018)
26. Lasser, D.J.: Topological ordering of a list of randomly-numbered elements of a network. Communications of the ACM **4**(4), 167–168 (1961)
27. Lee, W.C., Wu, C.C., Hsu, P.H.: A single-machine learning effect scheduling problem with release times. Omega **38**(1-2), 3–11 (2010)
28. Lodree, E.J., Geiger, C.D., Jiang, X.: Taxonomy for integrating scheduling theory and human factors: Review and research opportunities. International Journal of Industrial Ergonomics **39**(1), 39–51 (2009)
29. Lustig, I.J., Puget, J.F.: Program does not equal program: Constraint programming and its relationship to mathematical programming. Interfaces **31**(6), 29–53 (2001)
30. Neron, E., Artigues, C., Baptiste, P., Carlier, J., Damay, J., Demassey, S., Laborie, P.: Lower bounds for resource constrained project scheduling problem. In: Perspectives in modern project scheduling, pp. 167–204. Springer (2006)
31. Peteghem, V.V., Vanhoucke, M.: Influence of learning in resource-constrained project scheduling. Computers & Industrial Engineering **87**, 569–579 (2015)
32. Pritsker, A.A.B., Waiters, L.J., Wolfe, P.M.: Multiproject scheduling with limited resources: A zero-one programming approach. Management Science **16**(1), 93–108 (1969)
33. Qian, J., Steiner, G.: Fast algorithms for scheduling with learning effects and time-dependent processing times on a single machine. European Journal of Operational Research **225**(3), 547–551 (2013)
34. Rossi, F., Van Beek, P., Walsh, T.: Handbook of constraint programming. Elsevier (2006)
35. Schutt, A., Chu, G., Stuckey, P.J., Wallace, M.G.: Maximising the net present value for resource-constrained project scheduling. In: International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming. pp. 362–378. Springer (2012)
36. Schutt, A., Feydy, T., Stuckey, P.J., Wallace, M.G.: Explaining the cumulative propagator. Constraints **16**(3), 250–282 (2011)
37. Schwindt, C., Zimmermann, J., et al.: Handbook on Project Management and Scheduling. Springer (2015)
38. Van Peteghem, V., Vanhoucke, M.: Influence of learning in resource-constrained project scheduling. Computers & Industrial Engineering **87**, 569–579 (2015)
39. Vanhoucke, M., Debels, D.: The discrete time/cost trade-off problem: extensions and heuristic procedures. Journal of Scheduling **10**(4-5), 311–326 (2007)
40. Wei, C.M., Wang, J.B., Ji, P.: Single-machine scheduling with time-and-resource-dependent processing times. Applied Mathematical Modelling **36**(2), 792–798 (2012)
41. Yelle, L.E.: The learning curve: Historical review and comprehensive survey. Decision Sciences **10**(2), 302–328 (1979)
42. Zhu, G., Bard, J.F., Yu, G.: A branch-and-cut procedure for the multimode resource-constrained project-scheduling problem. INFORMS Journal on Computing **18**(3), 377–390 (2006)

Appendix A

In Table 2, we list the improved lower bounds that we obtain with the destructive lover bounding approach described in Subsection 4.3. Lower bounds marked with an asterisk (*) are proven optimal upper bounds.

Table 2. Improved lower bounds for 120-job PSPlib instances.

Instance	lb_{old}	lb_{new}
j1201_1.sm	104	105*
j1207_9.sm	84	85
j1208_6.sm	84	85*
j12012_7.sm	115	116
j12012_8.sm	110	111
j12013_10.sm	85	87
j12014_8.sm	108	109
j12019_4.sm	99	101
j12032_7.sm	117	118
j12032_9.sm	123	124
j12033_3.sm	100	101
j12034_5.sm	100	101
j12047_7.sm	111	112
j12059_9.sm	116	117