Learning Effect and Compound Activities in High Multiplicity RCPSP: Application to Satellite Production

Anh Duc, Stéphanie Roussel, Christophe Lecoutre, Anouck Chan

Learning Effect and Compound Activities in High Multiplicity RCPSP: Application to Satellite Production

Duc Anh Le[1]
DTIS, ONERA, Université de Toulouse, France

Stéphanie Roussel
DTIS, ONERA, Université de Toulouse, France

Christophe Lecoutre
CRIL, Université d'Artois & CNRS, France

Anouck Chan
DTIS, ONERA, Université de Toulouse, France

Abstract

This paper addresses the High Multiplicity Resource-Constrained Project Scheduling Problem (HM-RCPSP), in which multiple projects are performed iteratively while sharing limited resources. We extend this problem by integrating the learning effect, which makes the duration of some activities decrease when they are repeated. Learning effect can be represented by any decreasing function, allowing us to get flexibility in modeling various scenarios. Additionally, we take composition of activities into consideration for reasoning about precedence and resources in a more abstract way. A Constraint Programming model is proposed for this richer problem, including a symmetry-breaking technique applied to some activities. We also present a heuristic-based search strategy. The effectiveness of these solving approaches is evaluated through an experimentation conducted on data concerning real-world satellite assembly lines, as well as on some adapted literature benchmarks. Obtained results demonstrate that our methods serve as robust baselines for addressing this novel problem (denoted by HM-RCPSP/L-C).

2012 ACM Subject Classification Applied computing → Decision analysis

Keywords and phrases High-multiplicity Project Scheduling, Learning Effect, Compound Activities, Satellite Assembly Line, Constraint Programming, Symmetry Breaking

Digital Object Identifier 10.4230/LIPIcs.CP.2024.18

Supplementary Material *Dataset (PSP-based dataset)*: `https://doi.org/10.57745/ASGLBH` [18]

Funding *Stéphanie Roussel, Anouck Chan*: This work has been partly realised with the support of the French government, in the context of the BPI PSPC project "LiChIE" of the "Programme d'Investissements d'Avenir".
Christophe Lecoutre: This work has benefited from the support of the National Research Agency under France 2030, MAIA Project ANR-22-EXES-0009.

1 Introduction

In numerous industries, especially in manufacturing, a frequent challenge is to schedule several projects, each demanding multiple executions, within an environment of limited resources. This scheduling problem is known as the High Multiplicity Resource-Constrained Project Scheduling Problem (HM-RCPSP) [8].

[1] Corresponding author

This problem arises, for example, when planning the assembly of complex products such as satellites. Traditional satellite manufacturing tends to be handcrafted, and only a few units are produced over a long time horizon. Recently, there has been a growing interest in satellite constellations and large-scale satellite production is now essential to meet demand as highlighted in [7, 9]. We have been working on the production of Earth observation satellites, where most of the activities require human operators. In this case, satellite manufacturers have to adapt their industrial system to cope with very different numbers of units to be produced (from one to dozens) and different deadlines for these satellites. For example, it may be necessary to produce an Earth observation satellite every two months to keep up with constellation development, but a highly specialised sensor satellite can be produced in a dozen months. While each type of satellite has its own assembly process, the resources and machinery used to perform tasks are sometimes identical for satellites of different types. For instance, an Earth observation satellite and a highly dedicated sensor satellite both require aseismic areas in order to install instruments without any interference from the other tasks. As most of these resources and machines are generally very expensive, they are only available in limited quantities in the factory and are shared between the production of all satellite types.

Two features of the satellite assembly line scheduling problem cannot be modelled within the HM-RCPSP frame. Firstly, in addition to the classical precedence relationship over activities, one has also to deal with a composition relationship, which makes possible to express that an activity encompasses other ones (therefore starting and ending with the earliest and latest encompassed activity, respectively). Such an activity, called *compound*, can for instance be used for representing the "installation of a mirror on the satellite", which is decomposed into several atomic activities such as "fixing the mirror", "verifying the mirror alignment", etc. The interest for composition is twofold. It allows us to express the precedence relationship in a more compact way, and to indicate that a resource is consumed globally (the resource becoming unavailable for the whole duration of the compound activity). For instance, a bench test may be required for the overall mirror installation, expressing not only that each activity encompassed in this activity may use the bench test but also that the latter cannot be used by any other activity until the mirror installation is finished.

Secondly, in some manufacturing environments, the time required to perform an operation usually fluctuates over time. As a production system operates, workers acquire expertise or improve the manufacturing process, become proficient in required actions, learn tool utilization, and enhance their interaction with the supply chain, among other factors. Consequently, the overall system becomes more efficient. For example, the time needed to assemble the 10th iteration of a product may be only half of the time taken for the initial one. This phenomenon is recognized as the *learning effect*, and the variation in duration based on the repetition of the same operation is called a *learning curve* [25]. In the case of complex products such as satellites or aircrafts, such a phenomenon can have a significant impact on the medium and long-term production planning of the factory. Interestingly, taking learning effect into account permits a more realistic estimation of the dates at which products will be ready. It can also be useful to predict potential additional investment in terms of resources to meet delivery goals. However, incorporating learning effect makes HM-RCPSP more intricated because projects are inter-dependent in terms of activities duration.

The objective of this paper is to address the High Multiplicity RCPSP with Learning effect and Compound activities, denoted HM-RCPSP/L-C. More precisely, the contributions of this paper are:

- HM-RCPSP/L-C is formally defined, notably by introducing a generic learning effect component;
- a symmetry-breaking method is proposed, while proving its correctness;
- a Constraint Programming (CP) model and a heuristic-based search strategy are introduced;
- new academic benchmarks are proposed (and made publicly available);
- an experimentation is conducted on these benchmarks as well as on industrial satellite assembly line benchmarks, showing the potential of the CP approach (with and without symmetric-breaking constraints) which is compared to some other approaches.

The paper is structured as follows. After presenting some related works in Section 2, HM-RCPSP/L-C is formally introduced in Section 3. Then, symmetry breaking is addressed in Section 4. Two main solving approaches for HM-RCPSP/L-C are introduced in Section 5, and the results of the experiments we have performed are presented in Section 6. Finally, we conclude and discuss some perspectives in Section 7.

2 Related Works

The RCPSP problem is a classical problem in combinatorial optimization, for which plenty of solving approaches have been proposed in the literature [2]. As described in [12], many extensions have been proposed over the years. In this section, we first focus on extensions that address either the presence of several projects or their repetition in a context of shared resources. Then, we present some works in which scheduling and learning effects are simultaneously handled.

Multi Project and Repetition in Scheduling. In a recent survey [22], the authors present several extensions of the Resource-Constrained Multi-Project Scheduling Problem (RCMPSP). Such extensions can address activities features (e.g. preemption or uncertainty), activities relationships (e.g. concurrency or time lags), projects and resource features (e.g. renewability or availability). Because of the learning effect, the projects we consider in this study are inter-dependent, which prevents the HM-RCPSP/L-C to be seen as a special case of RCMPSP.

In [6], the authors address the cyclic RCPSP (a single project is repeated infinitely) through a Discrete Time Constraint Programming (DT-CP) approach. The objective is to find a pattern that interleaves several repetitions of the project in order to use resources optimally. In the context of construction projects [10], identical activities are repeated a given number of times. In [8], the authors address the HM-RCPSP/max in which several projects are repeated and share common resources. Projects can be linked through a generalized precedence relationship which specifies maximum time-lags between activities. They propose a symmetry breaking method for identical projects of the same class.

Learning Effect and Scheduling. The learning effect phenomenon was initially reported in [23]. The author defines a log-linear learning model that we use in our experiments. Various learning curves have been proposed since then [25].

There are two different approaches to learning in scheduling environments: (i) *position-based approach*, meaning that learning is effected by the pure number of times an activity has been completed; and (ii) *sum-of-processing-time approach* which takes into account the processing time of all same activities processed so far [5]. Some works have studied under which conditions the problem becomes polynomial [4, 3, 26]. In [1], a position-based scheduling problem with repetitive projects is studied using a two-stage approximation

approach. It consists of identical projects that can be executed in parallel while respecting the resource capacity, and each activity requires exactly one unit of one specific resource type.

Staff allocation has been studied in [24] and [21], respectively with a position based and a sum-of-processing based learning effect modelling. Learning effect has also been studied in the case of Discrete Time/Resource Trade-off Problem (DTRTP), as presented in [20, 19]. DTRTP is a sub-problem of the multi-mode RCPSP, where the duration of each activity depends on the amount of workers or resources assigned to it. Finally, in [13], the authors introduce four DT-CP formulations for the RCPSP with position-based learning effect, and provide an empirical comparison of their scheduling and lower bounding performance. In their study, the effect of learning is modeled by having two duration values for each activity: a nominal and a reduced one, and there is no repetition factor.

To conclude this section, to the best of our knowledge, there is no work in the literature addressing simultaneously multi-projects sharing resources, with compound activities and whose activity duration depends on learning effect.

3 HM-RCPSP/L-C

The components (data) required to define an instance of the high-multiplicity resource-constrained project scheduling problem with compound activities and learning effect (HM-RCPSP/L-C) are formally described in this section.

3.1 Problem Inputs

An instance of HM-RCPSP/L-C is a tuple composed of the following elements:

- $\mathcal{C}$ is the set of project classes (or categories);
- for each class $c \in \mathcal{C}$, $\mathcal{I}_c$ is the set of projects (or instances) of c that have to be realized. For each project $p \in \mathcal{I}_c$, due_p represents its due date, *i.e.* the date at which it should be finished;
- for each class $c \in \mathcal{C}$, $\mathcal{A}_c$ represents the set of non-preemptive activities that have to be realized for each project of c. Activities $\mathcal{A}_c$ can be partitioned into two sets $\mathcal{A}_c^A$ and $\mathcal{A}_c^C$ that respectively represent atomic activities and compound activities. Intuitively, a compound activity can be seen as a group of activities. Such an activity spans over all activities in the group;
- for each class $c \in \mathcal{C}$, $\mathcal{H}_c \subseteq \mathcal{A}_c^C \times \mathcal{A}_c$ is the composition relation of c. If a is a compound activity and b an activity, then $(a, b) \in \mathcal{H}_c$ expresses that b is encompassed by a (or a child activity of a). Note that it is possible to have (a, b) if b is a compound activity;
- for each class $c \in \mathcal{C}$ and for each atomic activity $a \in \mathcal{A}_c^A$, we assume the existence of a monotonically decreasing duration function, denoted as $dur_a : \mathbb{N}^+ \to \mathbb{N}^+$, which returns the duration of the activity based on the number of times it has been completely executed before;
- for each class $c \in \mathcal{C}$, $\mathcal{P}_c$ is a precedence relationship between activities in $\mathcal{A}_c$. If $(a, b) \in \mathcal{P}_c$, then activity a must be finished before the start of activity b;
- $\mathcal{R}$ is the set of cumulative and renewable resources. For each resource $r \in \mathcal{R}$, $capa_r$ denotes its capacity.
- for each resource $r \in \mathcal{R}$, for each class $c \in \mathcal{C}$, for each activity $a \in \mathcal{A}_c$, $cons_{r,a}$ denotes the amount of resource r consumed by a;
- $H \in \mathbb{N}^+$ is the maximum time horizon.

We denote a^p the instance of activity $a \in \mathcal{A}_c$ that has to be realized for project $p \in \mathcal{I}_c$ and $c \in \mathcal{C}$.

Assumptions. An instance of HM-RCPSP/L-C is said to be well-formed if and only if the following assumptions hold. Note that unless said otherwise, all instances considered in the paper are supposed to be well-formed.

1. Projects and activities are associated with exactly one class.
2. The duration function associated with each activity is monotonically decreasing. Formally, $\forall c \in \mathcal{C}, \forall a \in \mathcal{A}_c^A, \forall n \geq 0,\ dur_a(n+1) \leq dur_a(n)$.
3. For each class $c \in \mathcal{C}$, we suppose that the set of atomic activities $\mathcal{A}_c^A$ contains two virtual activities α_c and ω_c that respectively represent the start and end activities for a project. Such activities do not consume any resource, they are not encompassed by any compound activity and their duration is null. Moreover, α_c precedes (resp. ω_c follows) all other activities in $\mathcal{A}_c$.
4. For each class $c \in \mathcal{C}$, the graph induced by the compound/atomic activities relationship is a forest, *i.e.* each node has at most one parent and there is no cycle. Formally, such a graph $\mathcal{G}_c^{\mathcal{H}}$ is composed of a set of nodes $N_c^{\mathcal{H}}$ and a set of arcs $Arcs_c^{\mathcal{H}}$ that are defined as follows. $N_c^{\mathcal{H}}$ is composed of one node n_a for each activity a in $\mathcal{A}_c$. $Arcs_c^{\mathcal{H}}$ is composed of one arc (n_a, n_b) for each pair of activities $(a, b) \in \mathcal{H}_c$.
5. For each class $c \in \mathcal{C}$, for each tree in the forest $\mathcal{G}_c^{\mathcal{H}}$, and for each path from the root to an arbitrary leaf, the resource capacities are sufficient to execute all activities on this path simultaneously.
6. For each class $c \in \mathcal{C}$, the graph $\mathcal{G}_c^{\mathcal{P}}$ induced by the precedence relationship is acyclic. To build such a graph, we first define an atomic version of the precedence relationship, denoted $\mathcal{P}_c^A$, in which all precedences of $\mathcal{P}_c$ are transposed to atomic activities. Formally, for all a, b in $\mathcal{A}_c^A$, (a, b) belongs to $\mathcal{P}_c^A$ if and only if there exist two activities d and e in $\mathcal{A}_c$ such that: 1. $(d, e) \in \mathcal{P}_c$, 2. there exists a path in $\mathcal{G}_c^{\mathcal{H}}$ from n_d to n_a and 3. a path from n_e to n_b. Then, the set of nodes for $\mathcal{G}_c^{\mathcal{P}}$ is $N_c^{\mathcal{P}} = \{v_a \mid a \in \mathcal{A}_c^A\}$, that contains one node for each atomic activity. The set of arcs is $Arcs_c^{\mathcal{P}} = \{(v_a, v_b) \mid (a, b) \in \mathcal{P}_c^A\}$ and contains one arc for each pair in the atomic precedence relationship.

▶ **Example 1.** Figure 1 illustrates a toy example instance of HM-RCPSP/L-C. There are two classes, c_1 and c_2. The activities of each class are described on the left table of Figure 1a. For each project of class c_1, 4 activities (+2 dummy activities α_1 and ω_1) must be performed. Activities a, b, c, α_1 and ω_1 are atomic ($\mathcal{A}_{c_1}^A = \{\alpha_1, \omega_1, a, b, c\}$). Activity a's duration is given by function δ_1 described in the lower right corner: if no instance of activity a has been completed yet (i.e. $n = 0$), then its duration is 3, if one execution has been fully completed before starting a then its duration is equal to 2, and so on. Activity d is compound ($\mathcal{A}_{c_1}^C = \{d\}$) and its children are b and c, as represented on the upper graph of Figure 1b. This graph also represents the precedence relationship $\mathcal{P}_{c_1}$. For instance, (α_1, d) and (b, c) both belong to $\mathcal{P}_{c_1}$. r_1 and r_2 are the two resources and respectively have a capacity equal to 3 and 2. Both atomic and compound activities can consume these resources. For instance, $cons_{r_1,d} = 1$ and $cons_{r_2,b} = 1$. Two projects, p_1 and p_2, of class c_1 and one project, q, for class c_2 must be realized. Due dates for these projects are respectively equal to 7, 8 and 6.

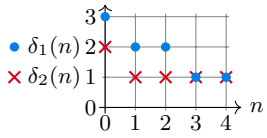
3.2 Schedule, Solution and Optimality

Schedule. A schedule σ for an HM-RCPSP/L-C instance is defined through the assignment of a start date to each atomic activity of each project of each class. Formally, if a is an atomic activity, the start date of a^p in σ is denoted $start^\sigma(a^p)$.

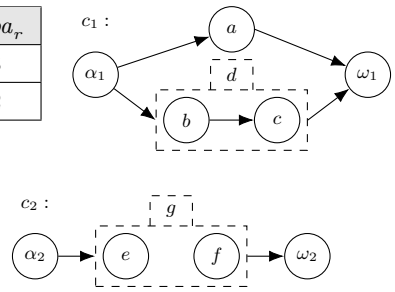
$\mathcal{C}$	$\mathcal{A}_c$	dur_a	r_1	r_2
c_1	a	δ_1	1	0
	b	δ_2	0	1
	c	δ_1	0	1
	d	-	1	0
c_2	e	δ_2	0	1
	f	δ_1	1	0
	g	-	1	0

$\mathcal{C}$	$\mathcal{I}_c$	due_p
c_1	p_1	7
	p_2	8
c_2	q	6

$\mathcal{R}$	$capa_r$
r_1	3
r_2	2



(a) Example of classes, projects, activities and resources.



(b) Composition and precedence relationships.

Figure 1 Toy example of a HM-RCPSP/L-C instance.

The end date of a^p, where a is an atomic activity and p a project, is denoted $end^\sigma(a^p)$. Because of the learning effect, a^p's duration depends on the number of times activity a has already been completed for other projects in the same class at the start of a^p. The function nc^σ, applied to activity a and time step t, returns the number of times that a has been completed at t. Formally,

- $end^\sigma(a^p) = start^\sigma(a^p) + dur_a(nc^\sigma(a, start^\sigma(a^p)))$.
- $nc^\sigma(a,t) = \sum_{q \in \mathcal{I}_c \setminus \{p\}} (end^\sigma(a^q) \leq t)$

In practice, the value of these functions can be determined for a given schedule σ by ordering the activities executions using their start dates. The start dates of two virtual activities α_c and ω_c can be then defined as follows:

- $start^\sigma(\alpha_c^p) = \min_{a \in \mathcal{A}_c^A} start^\sigma(a^p)$
- $start^\sigma(\omega_c^p) = \max_{a \in \mathcal{A}_c^A} end^\sigma(a^p)$

Start and end dates of compound activities can be deduced from those of atomic ones: the start date of a compound activity a (resp. the end date) is equal to the minimum start date (resp. maximum end date) of activities encompassed by a. For readability, we also use functions $start^\sigma(a^p)$ and $end^\sigma(a^p)$ for denoting start and end dates of compound activities.

Solution. A schedule σ is a solution for an HM-RCPSP/L-C instance if and only if the precedence constraints (Equation 1) are satisfied and the resources capacity is respected through the entire time horizon (Equation 2).

$$\forall c \in \mathcal{C}, \forall (a,b) \in \mathcal{P}_c, \forall p \in \mathcal{I}_c, \qquad end^\sigma(a^p) \leq start^\sigma(b^p) \tag{1}$$

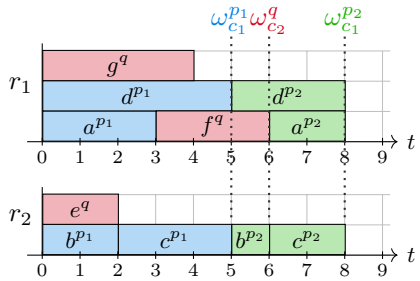
$$\forall r \in \mathcal{R}, \forall t \in [\![0, H]\!], \quad \left(\sum_{\substack{c \in \mathcal{C}, a \in \mathcal{A}_c, p \in \mathcal{I}_c \\ start^\sigma(a^p) \leq t < end^\sigma(a^p)}} cons_{r,a} \right) \leq capa_r \tag{2}$$

Optimal Solution. A solution σ is said to be optimal if it minimizes (in lexicographic order) the two following optimization criteria: (1) the sum of tardiness for projects (Equation 3), and (2) the makespan (Equation 4).

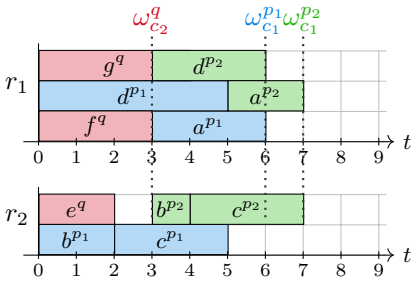
$$\sum_{c \in \mathcal{C}, p \in \mathcal{I}_c} \max\{0, end^\sigma(\omega_c^p) - due_p\} \tag{3}$$

$$\max_{c \in \mathcal{C}, p \in \mathcal{I}_c} end^\sigma(\omega_c^p) \tag{4}$$

▶ **Example 2.** Figures 2a and 2b respectively illustrate two solutions σ_1 and σ_2. We have $start^{\sigma}(c^{p_1}) = 2$, i.e. the execution of c in project p_1, starts at time 2 in solution σ_1. As $\delta_1(0) = 3$, c^{p_1} lasts 3 time units and $end^{\sigma}(c^{p_1}) = 5$. Still in σ_1, at the start of c^{p_2} (time step 6), one execution of c is completed so duration of c^{p_2} is equal to 2. Note that in σ_2, when starting activity c^{p_2}, the execution of c^{p_1} is not completed yet, so the duration of c^{p_2} is equal to 3.
As the compound activity g^q spans activities e^q and f^q, in solution σ_1, this makes g^q start at time 0 and end at time 6. However, in σ_2, g^q ends at time 3. In both cases, the resource r_1 is consumed all over g^q's duration.
Both solutions satisfy projects due dates. For instance, $end^{\sigma_1}(\omega_{c_1}^{p_1})$ is equal to 5, which is less than 7. However, the makespan of σ_2 is lower than the makespan of σ_1, which makes σ_2 a better solution.



(a) Solution σ_1 with a makespan equal to 8.



(b) Solution σ_2 with a makespan equal to 7.

Figure 2 Illustration of Example 2 – Due dates are satisfied in both solutions.

4 Symmetry Breaking

In this section, we establish the existence of symmetries between identical activities from different projects within the same class.

First, because of the learning effect, an activity a that starts in project q later than in project p can finish earlier in q. This is illustrated on the left part of Figure 3.

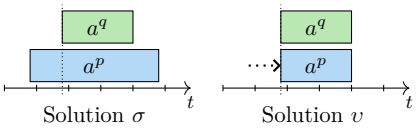


Figure 3 Illustration of Lemma 4's proof: σ is not start-end-consistent but υ is.

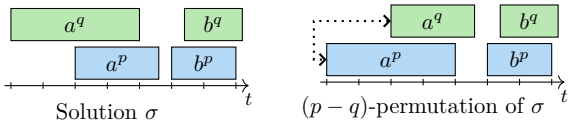


Figure 4 Illustration of a $(p-q)$-*permutation*: a^p and a^q are swapped.

We address that particular case through Definition 3 and Lemma 4.

▶ **Definition 3** (Start-end-consistency). *A solution σ is* start-end-consistent *if for any class $c \in \mathcal{C}$, any atomic activity $a \in \mathcal{A}_c^A$, and any two projects $p, q \in \mathcal{I}_c$, we have:*

$$start^{\sigma}(a^p) < start^{\sigma}(a^q) \iff end^{\sigma}(a^p) < end^{\sigma}(a^q)$$

▶ **Lemma 4.** *If a solution σ is not start-end-consistent then there exists a start-end-consistent solution υ that is equivalent or better than σ with respect to the two optimization criteria.*

Sketch of proof. Let a be an atomic activity and p and q be two projects for which a solution σ is not start-end-consistent as in Figure 3. We can build a schedule υ that is equal to σ except that the start date of a^p is delayed until the start date of a^q. All precedence and resource consumption constraints will be naturally satisfied in υ (they are relaxed). ◀

Note that the transformation for obtaining a start-end consistent solution (proof of Lemma 4) can delay some activities. However, as any activity delayed by the transformation also ends earlier, criteria can only be improved. From this point onward, we only consider start-end-consistent solutions. In the following, we define specific moves on schedules, that we call permutations, and show that they preserve the satisfaction of precedence and resource constraints.

▶ **Definition 5** ((p,q)-permutation). *For any solution σ, any class $c \in \mathcal{C}$ and any two projects $p, q \in \mathcal{I}_c$, a schedule π is a* (p,q)-permutation *of σ if for all atomic activities $a \in \mathcal{A}_c^A \setminus \{\alpha_c, \omega_c\}$, we have:*

- $\forall o \in \mathcal{I}_c \setminus \{p,q\}$, $start^\pi(a^o) = start^\sigma(a^o)$,
- *if* $start^\sigma(a^p) > start^\sigma(a^q)$, *then* $start^\pi(a^q) = start^\sigma(a^p)$ *and* $start^\pi(a^p) = start^\sigma(a^q)$,
- *else* $start^\pi(a^p) = start^\sigma(a^p)$ *and* $start^\pi(a^q) = start^\sigma(a^q)$.

The operation that consists in swapping start dates of a specific atomic activity, excluding the source and the sink, between two projects is called a swap.

As illustrated in Figure 4, a (p,q)-permutation consists in swapping all atomic activities of p that start after their counterpart activities in q. Because the start date of compound activities and source and sink depend only on the start dates of atomic activities, a corollary of Definition 5 is that all activities in p start before their counterpart activities in q. Note also that swapping the start dates of atomic activities a^p and a^q also swaps their duration and their end dates.

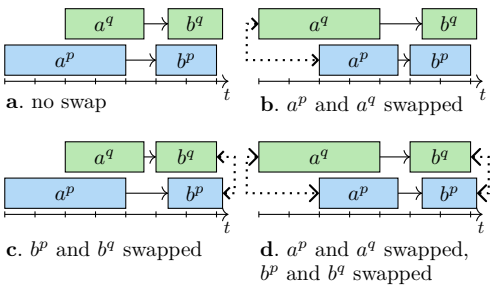
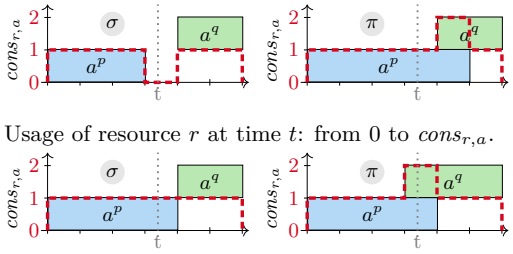


Figure 5 Four precedence cases considered in the proof of Lemma 6.



(a) Usage of resource r at time t: from 0 to $cons_{r,a}$.

(b) Usage of resource r at time t: from $cons_{r,a}$ to $2cons_{r,a}$.

Figure 6 Two impossible cases considered in proof of Lemma 7 when $a \in \mathcal{A}_c^C$.

▶ **Lemma 6.** *Given a start-end-consistent solution σ, for any class $c \in \mathcal{C}$ and any two projects $p, q \in \mathcal{I}_c$, the (p,q)-permutation π of σ is precedence-feasible, i.e. it satisfies Equation 1.*

Sketch of proof. As inferred by Assumption 5, we consider only precedence on atomic activities. If π is not precedence-feasible, there exist two atomic activities a and b such that a precedes b and such that $end^\pi(a^p) > start^\pi(b^p)$ or $end^\pi(a^q) > start^\pi(b^q)$ hold. As σ is precedence-feasible, then $end^\sigma(a^p) \leq start^\sigma(b^p)$ and $end^\sigma(a^q) \leq start^\sigma(b^q)$ both hold. We then consider all possible cases in terms of exchange, as illustrated in Figure 5, where the dashed lines correspond to an exchange of activities in π. In all cases, if σ is precedence-feasible, then so is π. ◀

▶ **Lemma 7.** *Given a start-end-consistent solution σ, for any class $c \in \mathcal{C}$ and any two projects $p, q \in \mathcal{I}_c$, the (p, q)-permutation π of σ is resource-feasible, i.e. it satisfies Equation 2.*

Sketch of proof. The resource feasibility is easy to prove in the case of atomic activities. In fact, when swapping activities, the consumption of resources does not change. However, in the case of compound activities, it is possible that some children activities are swapped and some other not, meaning that duration of compound activities can change when performing a (p, q)-permutation. In the following, we consider all the cases and show that even with compound activities, the permutation stays resource-feasible.

We define a function $\gamma_{r,a,t}(p, \sigma)$ that represents how much the activity a executed for project p consumes a resource r at a given time step t in σ, i.e. $\gamma_{r,a,t}(p, \sigma) = cons_{r,a}$ if $t \in T^{\sigma}_{p,a}$, 0 otherwise. The consumption of a resource r at each time step t in a schedule σ is given by the formula $\sum_{c \in \mathcal{C}, p \in \mathcal{I}_c, a \in \mathcal{A}_c} \gamma_{r,a,t}(p, \sigma)$. The proof is divided into several steps.

1. We show that if π is not-resource feasible, there exists a time-step t and a resource r such that $\sum_{c \in \mathcal{C}, p \in \mathcal{I}_c, a \in \mathcal{A}_c} \gamma_{r,a,t}(p, \pi) - \gamma_{r,a,t}(p, \sigma) > 0$. We consider this specific resource r and this time step t.
2. We prove that the resource capacity violation in π comes only from projects p and q.
3. Let c be the class associated with p and q and a an activity in $\mathcal{A}_c$, we show that the consumption of r induced by both activities a^p and a^q is equivalent in π and σ. Formally, $\gamma_{r,a,t}(p, \pi) - \gamma_{r,a,t}(p, \sigma) + \gamma_{r,a,t}(q, \pi) - \gamma_{r,a,t}(q, \sigma) = 0$.
   a. If a is an activity atomic, the usage of r unchanged.
   b. If a is a compound activity then we consider two cases (Figure 6): (i) if activities a^p and a^q do not use r at time t in σ, but at least one of them uses it in π, then we have $\gamma_{r,a,t}(p, \sigma) = \gamma_{r,a,t}(q, \sigma) = 0$ and $\gamma_{r,a,t}(p, \pi) + \gamma_{r,a,t}(q, \pi) \geq cons_{r,a}$ (Figure 5a) ; (ii) if exactly one activity (a^p or a^q) uses r at time t in σ and they both use it in π, then $\gamma_{r,a,t}(p, \sigma) + \gamma_{r,a,t}(q, \sigma) = cons_{r,a}$ and $\gamma_{r,a,t}(p, \pi) + \gamma_{r,a,t}(q, \pi) = 2cons_{r,a}$(Figure 5b). We show that both cases lead to a contradiction.

All cases imply a contradiction, meaning that π is resource-feasible. ◀

▶ **Lemma 8.** *Given a start-end-consistent schedule σ, for any class $c \in \mathcal{C}$ and any two projects $p, q \in \mathcal{I}_c$, the makespan of the (p, q)-permutation π of σ is equivalent to that of σ.*

Complete proof. Let $\epsilon^{\sigma}_{p,q} = \max_{a \in \mathcal{A}_c}(end^{\sigma}(a^p), end^{\sigma}(a^q))$ be the date at which both projects p and q are finished in σ. In π, even if there is a swap between activities, the date when p and q are both finished does not change, *i.e.* $\epsilon^{\pi}_{p,q} = \epsilon^{\sigma}_{p,q}$. As the timing of all other projects in π remains the same as in σ, the makespan of π must be equivalent to that of σ. ◀

For the tardiness, only some (p, q)-permutation such that $due_p < due_q$ allows to get a lower or equal tardiness. Therefore, we define a preorder on projects consistent with due dates and show that it preserves or improves tardiness. Note that we do not define a unique preorder $\prec$ based on uniquely considering $p \prec_c q$ when $due_p < due_q$, as we allow the user to set arbitrarily $p \prec_c q$ when $due_p = due_q$, provided that the relation remains well-formed. This is somewhat distantly related to the arbitrary choices made when posting the global constraint Precedence [17] (for example, between identical projects, you can choose one to precede the other, as done in [8]).

▶ **Definition 9** (Due-date consistent strict preorder). *For any class $c \in \mathcal{C}$, a strict order $\prec_c$ is due-date consistent for c if $\forall p, q \in \mathcal{I}_c$ such that $due_p < due_q$, then $p \prec_c q$.*
A relation $\prec$ is a due-date consistent strict preorder if it is the union of due-date consistent strict preorders $\prec_c$ for each class $c \in \mathcal{C}$.

▶ **Example 10.** Let us consider an instance with two classes, c_1 and c_2, and for which projects p_1, p_2 and p_3, and q_1 and q_2 are respectively associated, and such that $due_{p_1} = 7$, $due_{p_2} = due_{p_3} = 10$, $due_{q_1} = 8$ and $due_{q_2} = 6$. Then $\prec_{c_1} = \{(p_1, p_2), (p_1, p_3), (p_3, p_2)\}$ is a due-date consistent strict preorder for c_1. Considering (p_2, p_3) instead of (p_3, p_2) would also result in a due-date consistent strict preorder. $\prec_{c_2} = \{(q_2, q_1)\}$ is the only due-date consistent strict preorder for c_2. $\prec = \prec_{c_1} \cup \prec_{c_2}$ is a due-date consistent strict preorder for this instance.

In the following, for a class $c \in \mathcal{C}$, a project $p \in \mathcal{I}_c$ and a schedule σ, let τ_p^σ denote the tardiness of project p. Formally, $\tau_p^\sigma = \max\{0, end^\sigma(\omega_c^p) - due_p\}$. τ^σ denotes the sum of tardiness of all projects, as expressed in Equation 3.

▶ **Lemma 11.** *Given a start-end-consistent solution σ, a due-date consistent strict preorder $\prec$, a class $c \in \mathcal{C}$ and two projects $p, q \in \mathcal{I}_c$. If $p \prec q$ and π is the (p, q)-permutation of σ, then $\tau^\pi \leq \tau^\sigma$, i.e. the tardiness of π is less than or equal to the tardiness of σ.*

Sketch of proof. Except from projects p and q, all other projects activities remain the same in σ and π. Thus, the difference between tardiness of both schedules comes from the difference tardiness of projects p and q in σ and in π. Intuitively, p always ends earlier (its tardiness is reduced) but q might end later in π than in σ. We therefore have to show that $\tau_p^\pi + \tau_q^\pi \leq \tau_p^\sigma + \tau_q^\sigma$ or $\tau_q^\pi \leq \tau_q^\sigma$. We suppose that a^p and b^q are respectively the latest activities for p and q in σ. Cases considered in the following are illustrated in Figure 7.

1. If both activities a and b are not swapped between projects p and q in π, so b^q still concludes q in π, which implies $\tau_q^\pi = \tau_q^\sigma$.
2. If a^p and a^q are swapped, but b^p and b^q are not then $\tau_p^\pi \leq \tau_p^\sigma$.
   a. If $end^\sigma(a^p) \leq end^\sigma(b^q)$ then we can show that b^q still concludes q in π, so $\tau_q^\pi = \tau_q^\sigma$.
   b. Else, a^q concludes q in π and we have to consider several following situations.
      i. If $end^\sigma(a^p) \leq due_q$, then $\tau_q^\pi = \tau_q^\sigma = 0$.
      ii. If $end^\sigma(a^p) > due_q$, then $\tau_q^\pi = end^\sigma(a^p) - due_q$ and $\tau_p^\sigma = end^\sigma(a^p) - due_p$. Regardless of the relative positions of $end^\sigma(b^q)$, due_p and due_q, we show that $\tau_p^\pi + \tau_q^\pi \leq \tau_p^\sigma + \tau_q^\sigma$.
3. It is impossible to have b^p and b^q being swapped but not a^p and a^q, because a^p and b^q are respectively the latest activities for p and q in σ.
4. If both a^p and a^q are swapped, and so are b^p and b^q, then $end^\sigma(a^q) \leq end^\sigma(b^q) < end^\sigma(b^p) \leq end^\sigma(a^p)$ then b^p and a^q are respectively the latest activities for p and q in π. We prove that $\tau_p^\pi + \tau_q^\pi \leq \tau_p^\sigma + \tau_q^\sigma$ in several cases: a. both p and q are late in σ, b. p is late and q is early in σ and c. both p and q are early in σ.

In those all cases, $\tau_p^\pi + \tau_q^\pi \leq \tau_p^\sigma + \tau_q^\sigma$, which proves that $\tau^\pi \leq \tau^\sigma$. ◀

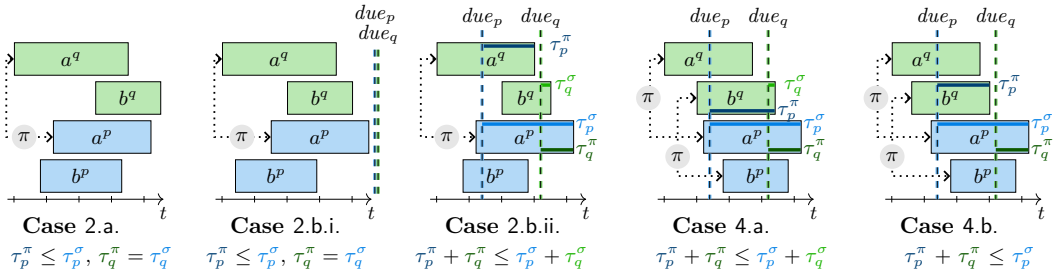


Figure 7 Illustration of cases considered in Lemma 11's proof.

We now define a $\prec$-permutation of σ in which (p,q)-permutations are done for all p,q such that $p \prec q$ and show that if σ is a solution then the $\prec$-permutation is also a solution better or equivalent to σ.

▶ **Definition 12** ($\prec$-permutation). *Given a solution σ and a due-date consistent strict preorder $\prec$, a schedule π is a $\prec$-permutation of σ if, for any class $c \in \mathcal{C}$, and any two projects p, q in $\mathcal{I}_c$ such that $p \prec q$, π is a (p,q)-permutation of σ.*

▶ **Lemma 13.** *Given a start-end-consistent solution σ and a due-date consistent strict preorder $\prec$, the $\prec$-permutation π of σ is a solution with a makespan equivalent to that of σ and a tardiness as least as good as that of σ.*

Complete proof. Let π be a $\prec$-permutation of a schedule σ. π can be built from σ through several schedules $\upsilon_0, \ldots, \upsilon_n$ where $\upsilon_0 = \sigma$, $\upsilon_n = \pi$ and $\forall i \in [\![1,n]\!]$, υ_i is a (p,q)-permutation of υ_{i-1} with p and q two projects such that $p \prec q$. According to Lemmas 6, 7, 8 and 11, each of these permutations is a solution with an equivalent makespan and equivalent or lesser tardiness. ◀

This result means that for finding a solution σ to an HM-RCPSP/L-C instance, one may consider a due date consistent preorder $\prec$ and look for solutions in which for all p and q such that $p \prec q$, for all activity $a \in \mathcal{A}_c$, $start^\sigma(a^p) \leq start^\sigma(a^q)$.

5 Solving HM-RCPSP/L-C

In this section we present the two main approaches we have developed for solving HM-RCPSP/L-C, namely a CP approach and a heuristic-based search approach. They can be combined, but the exact description of these combinations is left to Section 6.

5.1 Building a CP Model

Regarding the CP approach, in this paper, we use the interval based variable formulation of Optimization Programming Language (OPL) associated with CP Optimizer [16]. An interval variable allows us to represent an activity along with its variable starting date, its variable duration, its mandatory or optional presence in the computed schedule, its earliest start and latest end dates. The start date, end date and duration of an interval variable are respectively accessible through functions $\mathsf{startOf}$, endOf and $\mathsf{lengthOf}$. For each class $c \in \mathcal{C}$, for each project $p \in \mathcal{I}_c$, for each activity $a \in \mathcal{A}_c$, we consider the following variables:

- $\mathbf{itv}_{a,p} \in [\![0,H]\!]$ is an interval variable for the execution of activity a^p;
- $\mathbf{n}_{a,p} \in [\![0,|\mathcal{I}_c|]\!]$ is an integer variable representing the number of times activity a of class c was completed before the start of this instance of a, corresponding to $nc^\sigma(a, start^\sigma(a^p))$ defined in Section 3.

Constraints and criteria are encoded as follows:

$$minimize \left(\sum_{\substack{c\in\mathcal{C},p\in\mathcal{I}_c \\ a\in\mathcal{A}_c}} \max(0, \mathsf{endOf}(\mathbf{itv}_{a,p}) - due_p), \max_{\substack{c\in\mathcal{C},p\in\mathcal{I}_c \\ a\in\mathcal{A}_c}} \mathsf{endOf}(\mathbf{itv}_{a,p}) \right) \tag{5}$$

such that:

$$\forall r \in \mathcal{R}, \quad \textstyle\sum_{\substack{c\in\mathcal{C},p\in\mathcal{I}_c, \\ a\in\mathcal{A}_c}} \mathsf{pulse}(\mathbf{itv}_{a,p}, cons_{r,a}) \leq capa_r \tag{6}$$

$$\forall c \in \mathcal{C}, \forall p \in \mathcal{I}_c, \forall (a,b) \in \mathcal{P}_c, \quad \mathsf{endBeforeStart}(\mathbf{itv}_{a,p}, \mathbf{itv}_{b,p}) \tag{7}$$

$$\forall c \in \mathcal{C}, \forall p \in \mathcal{I}_c, \forall a \in \mathcal{A}_c^C, \quad \mathsf{span}(\mathbf{itv}_{a,p}, \{\mathbf{itv}_{b,p} | (a,b) \in \mathcal{H}_c\}) \tag{8}$$

$$\forall c \in \mathcal{C}, \forall p \in \mathcal{I}_c, \forall a \in \mathcal{A}_c^A, \quad \mathbf{n}_{a,p} = \textstyle\sum_{q\in\mathcal{I}_c} (\mathsf{endOf}(\mathbf{itv}_{a,q}) \leq \mathsf{startOf}(\mathbf{itv}_{a,p})) \tag{9}$$

$$\forall c \in \mathcal{C}, \forall p \in \mathcal{I}_c, \forall a \in \mathcal{A}_c^A, \quad \mathsf{lengthOf}(\mathbf{itv}_{a,p}) = dur_a(\mathbf{n}_{a,p}) \tag{10}$$

The lexicographic order of criteria is formalized in Equation (5). Constraints (6) are cumulative global constraints with respect to resource capacities, expressed through the OPL pulse function. Constraints (7) guarantee that the precedence relationships are satisfied (OPL function endBeforeStart). Constraints (8) enforce compound activities to span over their children. Constraints (9) encode the computation of $nc^{\sigma}(a,)$ at the time the atomic activity a^p starts. The duration with learning effect are pre-computed and atomic activities are assigned their duration through *Element* Constraints (10). In addition to the original constraints mentioned above, symmetry breaking constraints are defined as follows:

$$\forall c \in \mathcal{C}, \forall p, q \in \mathcal{I}_c \ s.t. \ p \prec_c q, \forall a \in \mathcal{A}_c^A, \mathsf{startBeforeStart}(\mathbf{itv}_{a,p}, \mathbf{itv}_{a,q}) \tag{11}$$

5.2 Using Squeaky Wheel Optimization

The second main approach we use to address HM-RCPSP/L-C is called Squeaky Wheel Optimisation (SWO) [14]. This technique consists of constructing an initial solution using a greedy algorithm. It is followed by an analysis to identify areas for improvement, i.e. parts of the solution that might improve the objective function score if they are modified. From this analysis, new priorities are generated, which modifies the sequence in which the greedy algorithm constructs subsequent solutions. This iterative process is repeated until a predefined limit is reached.

More precisely, the greedy approach for constructing a solution of HM-RCPSP/L-C consists in a Parallel Schedule Generation Scheme. Starting with an empty schedule, it iterates chronologically over the time horizon and fills the schedule with eligible activities. For a given time step t, an atomic activity a is eligible if Equations (1) and (2) are satisfied when a starts at time t (taking the compound activity of a into account). Using a choice heuristic, an eligible activity is inserted at time t and the set of eligible activities is updated. When this set becomes empty, the process is repeated for the next time step $t+1$. The schedule is complete when all activities have been inserted. Given a set of eligible activities $\mathcal{E}$, the choice heuristic follows several steps.

1. Because of the symmetry proof, we first remove from $\mathcal{E}$ all the activities a^q for which there exists a project p of the same class as q such that p precedes q in the due-date consistent strict preorder ($p \prec q$) and a^p has not been inserted yet ($a^p \in \mathcal{E}$). The resulting set of eligible activities is denoted $\mathcal{E}_{SB}$.
2. There are two possible cases: a) if there is no late project at the moment, all available projects with activities in $\mathcal{E}_{SB}$ will be considered as candidates; b) otherwise, only the late projects will be considered as candidates. For each candidate project p of a class c, the *remaining-graph* of p is defined as follows: i) for each $a \in \mathcal{A}_c^A$ which is already inserted, we remove from $\mathcal{P}_c^A$ the vertex a and all the arcs related to a; ii) the weight of an arc (a,b) in the remaining-graph is the duration of activity a. Each candidate project p is assigned a probability based on the ratio $CPL'_p/(due_p - t)$ if p is early and $1/CPL'_p$ otherwise, where CPL'_p is the critical path length of the remaining-graph of p. One project is then selected according to the resulting probability distribution. The set of eligible activities is reduced to $\mathcal{E}_{SB}^p$ by considering only activities from the selected project p.
3. An activity is randomly selected in $\mathcal{E}_{SB}^p$ using a uniform probability distribution.

As the heuristic choice is stochastic, the greedy algorithm is run several times until a given number of solutions have been constructed without improvement. At each step, the tardiness of the projects in the best-constructed solution is analysed to modify the probability of projects for the next cycle. To achieve that, we increase (resp. decrease) the due date of the most advanced (resp. the tardiest) project: this corresponds to a random portion of

the smallest amount between the earliness of the most advanced project and the due date of the tardiest project (in our datasets, we do not have the case in which all projects are late). Note that these modified due dates are only used to calculate the project selection probability, not the tardiness objective. This iterative cycle continues until a given timeout condition is reached, and the best-found solution is returned.

6 Experimentation

In this section, we present the results of some experiments conducted on real-world and synthetic benchmarks. Before presenting the experimental results, we describe the benchmarks and the solving approaches tested, including the main ones and their combinations or derivations.

6.1 Benchmarks

Learning Curves. In this research, we use the log-linear learning curve with a steady learning state [23, 25, 11][2]. Formally, for each class $c \in \mathcal{C}$, each atomic activity $a \in \mathcal{A}_c^A$ and each integer n, we consider the following elements: (i) the duration for the first execution, denoted dur_a^0; (ii) the steady state of learning, denoted dur_a^∞, that represents the ultimate duration of a; (iii) the learning effect $l_a \in]0, 1]$, that impacts the duration curve slope. Then, for all $n \geq 0$, $dur_a(n) = dur_a^\infty + \lceil (dur_a^0 - dur_a^\infty) \cdot (n+1)^{\log_2 l_a} \rceil$. In our experiments, we consider that $dur_a^\infty = \frac{1}{2} dur_a^0$.

Table 1 Features of instances per dataset: number of classes, atomic activities, compound activities, projects, resources, capacity of resources, and learning rate.

Instances		$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{A}_c\rvert$	$\lvert\mathcal{A}_c^C\rvert$	$\lvert\mathcal{I}_c\rvert$	$\lvert\mathcal{R}\rvert$	$capa_r$	l_a
PSP-based	small	$[\![2, 4]\!]$	$\{30, 60\}$	0	$[\![5, 10]\!]$	4	$\sum_{i_{LIB}} capa_r^{i_{LIB}}$	$[0.45, 0.95]$
	large	$[\![5, 7]\!]$	$\{60, 90, 120\}$					
Satellite	original	3	≤ 30	≤ 3	≤ 5	40	≤ 16	$[0.05, 0.95]$
	extended	6			≤ 14			0.85

Datasets. We tested our approaches on two datasets presented in Table 1. First, we created the PSP-based dataset (50 instances), where each class is an RCPSP instance from PSPLIB [15]. The due dates are computed as follows: a project has a due date equal to the maximum due date d in the original PSPLIB instance, then for each other project's due date, we iteratively add $k \times d$, where k is a random value in $[0.3, 0.8]$. The time horizon is equal to 40 (resp. 70) times the largest due dates among the involved PSPLIB instances in small (resp. large) instances. The capacity of each resource is the sum of the capacities of that resource from the involved PSPLIB instances. This dataset does not include any compound activity. This dataset is publicly available ([18]).

The satellite dataset contains 24 instances updated from a satellite manufacturer. The original set of instances, described in Table 1, covers a time horizon of more than one year with activities lasting from a few hours to a few days. We only vary the learning rate in these instances. We have created larger instances by increasing the number of classes, projects,

[2] This curve was not provided by our industrial partner but is classically used in the aerospace domain.

and the horizon from the original data while maintaining the resource capacities. The due dates for the additional projects have been randomly selected with good diversity: the due dates are distributed so that there are at most two satellites in every thousand time unit.

6.2 Solving Approaches

We have compared various approaches, based on the main techniques presented in Section 5.

CP-based Approaches. Two CP (variants of) models of HM-RCPSP/L-C are defined as follows: i) LC, Learning Curve, corresponding to Constraints (5) to (10), and ii) LCSB, Learning Curve and Symmetry Breaking, including Constraints (11) as well.

SWO Approach. In our implementation, the greedy algorithm is run until the solution has not been improved 150 times in a row.

Hybrid Approach. SWO is able to produce solutions in a very short time, whereas CP-based approaches can take longer to find a first "good" solution. Therefore, we have tested an hybrid approach, denoted HYB, which consists of computing a solution with SWO in a short time (5 seconds in the experiments) and then using that solution as a starting point for LCSB (that tries to improve it for the rest of the time).

Constant then Learning Curve Approach. The Constant then Learning Curve approach, denoted CLC, aims to adress the complexity of dealing with varying durations (due to learning curves). Such an approach consists of the following steps:

1. we consider the Constant Duration variant of LCSB, denoted CD, in which Constraints (10) are replaced by constraints imposing a constant duration for all activities (actually solving HM-RCPSP/C);
2. we extract the resource-accessing order of activities from the CD solution, i.e. the execution order of all activities;
3. using a Parallel Generation Scheme, we chronologically compute the start date of each activity using the fixed execution order of step 2., considering that the durations of activities follow the specified learning curves.

We have tested three variations of this approach, denoted CLC^0, CLC^{mid} and CLC^∞, depending on the value chosen as constant duration in CD: this value is respectively dur_a^0, $1/2(dur_a^0 + dur_a^\infty)$ and dur_a^∞. For each dataset, only the results of the variation with the best performance will be presented in the main paper result tables (complete results for the PSPLib-based dataset are available online [18]).

6.3 Experimental Results

The tests have been launched using IBM CP Optimizer 22.1.1 through the DOcplex API for the CP-based approaches and Julia v1.10.2 for the SWO-based approaches, on Intel® Xeon® CPU E5-2660v3 2.60-3.30 GHz with 62 GB of RAM. All approaches have been launched twice, to compare them over a short timeout (15 seconds) and a long timeout (2 hours).

Satellite Dataset. For the original satellite instances, all approaches achieve zero tardiness within 15 seconds, except CLC^{mid} and CLC^∞. The final solutions obtained by CP-based approaches after 2 hours are the same as those obtained after 15 seconds. Interestingly, LCSB and HYB have successfully proved the optimality of 2 instances after 2 hours (whereas

SWO has the worst makespan values). For the five extended satellite instances, the tardiness values are much higher as showed in Table 2. In the 2-hour test, LCSB clearly outperforms the others in 4 instances out of 5. LC falls behind and struggles to order projects in a way that minimizes tardiness. LCSB also provides the best makespan values in all instances but one (not visible in Table 2). Note that the optimality was not proved by any approach. In the 15-second test, HYB achieves the best tardiness in more than half of the instances, but CLC^{mid} surprisingly performs best overall.

Table 2 Tardiness values obtained by all solving approaches on the extended satellite dataset.

$\sum_{c,p} \lvert\mathcal{A}_c\rvert$	2 hours					15 seconds				
	LC	LCSB	SWO	HYB	CLC^{mid}	LC	LCSB	SWO	HYB	CLC^{mid}
637	1,341	1,318	4,566	**1,243**	1,668	5,516	3,286	6,979	**2,021**	2,033
913	1,267	**1,210**	5,963	**1,210**	2,443	9,906	9,981	6,734	**2,544**	2,582
1,217	2,004	**1,786**	13,174	2,626	4,372	48,099	22,121	21,854	**12,510**	13,893
1,526	10,993	**2,135**	23,530	2,149	5,132	126,012	41,409	33,228	41,204	**17,430**
1,832	13,372	**3,040**	39,440	4,403	8,268	210,673	30,827	54,347	54,266	**26,979**

PSP-based Dataset. In Table 3, we present for each criterion and each approach the number of times the best value is computed (obtained) compared to others, and also the average difference from the best value found by all approaches, denoted ADBV, which can be calculated by: $\text{ADBV}(obj, \lambda) = \sum_{i \in \mathcal{I}} (obj_i^{\lambda} - obj_i^{best}) / \sum_{i \in \mathcal{I}} (obj_i^{\lambda} \neq obj_i^{best})$, where $\mathcal{I}$ is the set of instances, obj_i^{λ} is the final value of the objective obj (either tardiness or makespan) found by approach λ in instance i, and obj_i^{best} is the best-found value of the objective obj for instance i by all approaches.

Table 3 Number of times the best value (#Best) is obtained and ADBV values for all solving approaches on the PSP-based dataset.

Timeout	Dataset	Objective	#Best					ADBV				
			LC	LCSB	SWO	HYB	CLC^0	LC	LCSB	SWO	HYB	CLC^0
2h	small	Tardiness	**21**	**21**	0	**21**	15	**13**	27	144	28	113
		Makespan	18	**19**	0	14	0	3	**1**	14	2	31
	large	Tardiness	**17**	16	0	16	7	**66**	109	629	72	451
		Makespan	14	15	4	**16**	0	7	**6**	10	7	62
15s	small	Tardiness	**22**	16	0	19	15	48	40	126	**38**	115
		Makespan	**16**	15	2	10	0	4	**3**	12	5	33
	large	Tardiness	**15**	4	6	3	5	5,494	–	386	**307**	2,011
		Makespan	8	2	6	**12**	0	23	–	**7**	10	96

In the 2-hour test, for the PSP-based instances, LC, LCSB, and HYB performed best, successfully meeting all due dates in 31 to 33 out of 50 instances (not visible in Table 3). Nevertheless, LCSB has a slight advantage when considering the second criterion (makespan). CLC^0 performs satisfactorily but it has a poor ADBV value of tardiness. Clearly, SWO has the worst performance since it never reaches the best value. Note that the optimality was not proved by any solving approach regardless the dataset (small or large).

In the 15-second test, the three approaches LC, LCSB, and HYB keep taking the lead for the small PSP-based dataset, with a slight advantage for LC. However, for the large dataset, the pure CP-based approach LC sometimes struggles to find a good solution in time, resulting in a large ADBV value, even though it achieves the best tardiness in 15 out of 24 instances. SWO showcases its strengths when dealing with big-size instances in a low-timeout condition, while LCSB cannot find a solution for half of the instances. The hybrid approach HYB shows the best overall performance in the 15-second test, with the lowest ADBV value across both small and large PSP-based datasets.

7 Conclusion

In this paper, we have i) formally defined the High Multiplicity RCPSP with Compound activities and Learning effect, ii) proposed a CP model for this problem, iii) proved the existence of symmetric projects within each class, and iv) adapted a heuristic-based search (SWO) using the proof of symmetric projects. Additionally, we compared the performance of various solving approaches using an industrial satellite dataset and a PSP-based one.

On satellite assembly instances, the approaches presented in this paper can generate a schedule within a reasonable amount of time. Interestingly enough, we have observed that the performance of the CP-based approaches was boosted by breaking symmetries on these industrial instances. Importantly, built schedules allow satellite engineers to effectively scale resources, particularly human means, and anticipate potential delays. Besides, industrial partners will be able to fine-tune the schedules according to their environment learning effect features, as our approach is completely generic on that point.

For future research, it would be beneficial to explore cases where learning effect is shared among similar activities across different classes, which will render projects from different classes interdependent. Additionally, considering the uncertainty in learning efficiency could be explored to enhance the robustness of the schedule.

References

1 J.P. Amor. Scheduling programs with repetitive projects using composite learning curve approximations. *Project Management Journal*, 33(3):16–29, 2002.

2 C. Artigues, S. Demassey, and E. Neron. *Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications*. ISTE/Wiley, 2008. URL: `https://hal.science/hal-00482946`.

3 A. Bachman and A. Janiak. Scheduling jobs with position-dependent processing times. *Journal of the Operational Research Society*, 55(3):257–264, 2004.

4 D. Biskup. Single-machine scheduling with learning considerations. *European Journal of Operational Research*, 115(1):173–178, 1999.

5 D. Biskup. A state-of-the-art review on scheduling with learning effects. *European Journal of Operational Research*, 188(2):315–329, 2008.

6 A. Bonfietti, M. Lombardi, L. Benini, and M. Milano. A constraint based approach to cyclic RCPSP. In *Proceedings of CP'11*, pages 130–144. Springer, 2011.

7 C. Daehnick, I. Klinghoffer, B. Maritz, and B. Wiseman. Large LEO satellite constellations: Will it be different this time? *McKinsey & Company*, 4, 2020.

8 S.J. Edwards, D. Baatar, K. Smith-Miles, and A.T. Ernst. Symmetry breaking of identical projects in the high-multiplicity rcpsp/max. *Journal of the Operational Research Society*, 72(8):1822–1843, 2021.

9 M. Eugeni, T. Quercia, M. Bernabei, A. Boschetto, F. Costantino, L. Lampani, A. Marchetti Spaccamela, A. Lombardo, M. Mecella, L. Querzoni, R. Usinger, M. Aliprandi, A. Stancu, M.M. Ivagnes, G. Morabito, A. Simoni, A. Brandão, and P. Gaudenzi. An industry 4.0 approach to large scale production of satellite constellations. the case study of composite sandwich panel manufacturing. *Acta Astronautica*, 192:276–290, March 2022. doi:10.1016/j.actaastro.2021.12.039.

10 J.D. García-Nieves, J.L. Ponz-Tienda, A. Ospina-Alvarado, and M. Bonilla-Palacios. Multipurpose linear programming optimization model for repetitive activities scheduling in construction projects. *Automation in Construction*, 105:102799, 2019. doi:10.1016/j.autcon.2019.03.020.

11 E.H. Grosse, C.H. Glock, and S. Müller. Production economics and the learning curve: A meta-analysis. *International Journal of Production Economics*, 170:401–412, 2015.

12 S. Hartmann and D. Briskorn. An updated survey of variants and extensions of the resource-constrained project scheduling problem. *Eur. J. Oper. Res.*, 297(1):1–14, 2022. doi:10.1016/J.EJOR.2021.05.004.

13 A. Hill, J. Ticktin, and T.W.M. Vossen. A computational study of constraint programming approaches for resource-constrained project scheduling with autonomous learning effects. In *Proceedings of CPAIOR'21*, pages 26–44. Springer, 2021.

14 D.E. Joslin and D.P. Clements. Squeaky wheel optimization. *Journal of Artificial Intelligence Research*, 10:353–373, 1999.

15 R. Kolisch and A. Sprecher. Psplib - a project scheduling problem library: Or software - orsep operations research software exchange program. *European Journal of Operational Research*, 96(1):205–216, 1997. doi:10.1016/S0377-2217(96)00170-1.

16 P. Laborie, J. Rogerie, P. Shaw, and P. Vilím. Ibm ilog cp optimizer for scheduling: 20+ years of scheduling with constraints at ibm/ilog. *Constraints*, 23:210–250, 2018.

17 Y.C. Law and J. Lee. Global constraints for integer and set value precedence. In *Proceedings of CP'04*, pages 362–376, 2004.

18 D.A. Le and S. Roussel. Dataset for the High Multiplicity RCPSP with Learning Effect and Compound Activities, 2024. Dataset, version 3.2. (visited on 2024-08-19). doi:10.57745/ASGLBH.

19 V. Van Peteghem and M. Vanhoucke. A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, 201(2):409–418, 2010.

20 V. Van Peteghem and M. Vanhoucke. Influence of learning in resource-constrained project scheduling. *Computers & Industrial Engineering*, 87:569–579, 2015.

21 S. Qin, S. Liu, H. Kuang, et al. Piecewise linear model for multiskilled workforce scheduling problems considering learning effect and project quality. *Mathematical problems in Engineering*, 2016, 2016.

22 M. Gómez Sánchez, E. Lalla-Ruiz, A. Fernández Gil, C. Castro, and S. Voß. Resource-constrained multi-project scheduling problem: A survey. *European Journal of Operational Research*, 309(3):958–976, 2023. doi:10.1016/j.ejor.2022.09.033.

23 T.P. Wright. Factors affecting the cost of airplanes. *Journal of the aeronautical sciences*, 3(4):122–128, 1936.

24 M.C. Wu and S.H. Sun. A project scheduling and staff assignment model considering learning effect. *The International Journal of Advanced Manufacturing Technology*, 28:1190–1195, 2006.

25 L.E. Yelle. The learning curve: Historical review and comprehensive survey. *Decision sciences*, 10(2):302–328, 1979.

26 Y. Yin, D. Xu, K. Sun, and H. Li. Some scheduling problems with general position-dependent and time-dependent learning effects. *Information Sciences*, 179(14):2416–2425, 2009.

Appendix

In this appendix, we provide the complete proofs of lemmas in Section 4 for breaking the symmetry. Note that we believe that sketches of proof provided in the paper are sufficient for understanding and reproducing the complete ones.

Note that the complete result tables of the tests in Section 6 are available at the datasets URL.

A Lemma 4

Complete proof. Let σ be a solution that is not start-end-consistent. There exists a class $c \in \mathcal{C}$, an atomic activity $a \in \mathcal{A}_c^A$, two projects p and q in $\mathcal{I}_c$ such that $start^\sigma(a^p) < start^\sigma(a^q)$ and $end^\sigma(a^p) \geq end^\sigma(a^q)$. We create a new solution $\upsilon_{a,p,q}$ such that for all activities b in $\mathcal{A}_c^A$, for all projects $u \in \mathcal{I}_c$ such that $b^u \neq a^p$, $start^\sigma(b^u) = start^{\upsilon_{a,p,q}}(b^u)$. Then, for activity a and project p, we delay the start of activity a^p in order to make it start at the same time as a^q. Formally, we have $start^{\upsilon_{a,p,q}}(a^p) = start^{\upsilon_{a,p,q}}(a^q) = start^\sigma(a^q)$. We also have $end^{\upsilon_{a,p,q}}(a^p) = end^{\upsilon_{a,p,q}}(a^q) = end^\sigma(a^q)$.

As a^p starts later and finishes earlier in $\upsilon_{a,p,q}$, all precedence that are satisfied in σ are also satisfied in $\upsilon_{a,p,q}$. The same holds for the resource consumption. If a^p is the activity which concludes project p in σ (*i.e.* $end^\sigma(\omega_c^p) = end^\sigma(a^p)$), then we have $end^{\upsilon_{a,p,q}}(a^p) \leq end^{\upsilon_{a,p,q}}(\omega_c^p) \leq end^\sigma(\omega_c^p)$. The end date of project p in $\upsilon_{a,p,q}$ either becomes earlier or remains the same as in σ. As the timing for all other projects remains the same as in σ, the values of both criteria for $\upsilon_{a,p,q}$ are either better than or equivalent to σ.

We perform the solution modification iteratively for each tuple (a, p, q) that is not start-end-consistent. The resulting solution υ is start-end-consistent and at least as good as σ for both criteria. ◀

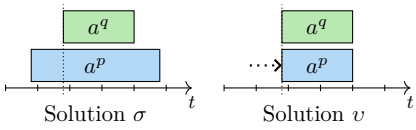


Figure 8 Illustration of Lemma 4's proof: σ is not start-end-consistent but υ is.

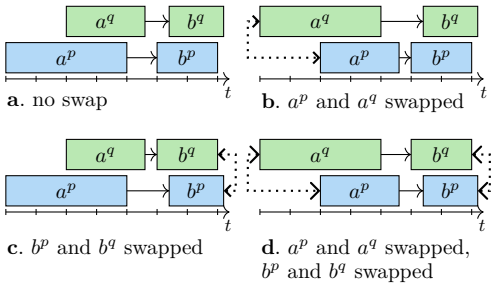


Figure 9 Four precedence cases considered in the proof of Lemma 6.

B Lemma 6

Complete proof. We assume that π is not precedence-feasible. Since all precedence relations in $\mathcal{P}_c$ can be transposed into $\mathcal{P}_c^A$ (see Assumption 5), which is concerned only with atomic activities, this implies the existence of two atomic activities $a, b \in \mathcal{A}_c^A$ such that $(a, b) \in \mathcal{P}_c^A$ and such that at least one of the following equations hold:

$$end^\pi(a^p) > start^\pi(b^p) \tag{12}$$

$$end^\pi(a^q) > start^\pi(b^q) \tag{13}$$

Due to the definition of the start dates for the two virtual activities α_c and ω_c (see Schedule), these activities never contribute to a precedence violation and they are not considered in the following.

Case 1. $start^\sigma(a^p) \leq start^\sigma(a^q)$ and $start^\sigma(b^p) \leq start^\sigma(b^q)$ (Figure 9a). This means that there is no swap of activities, and the start dates of a^p, a^q, b^p, b^q in π remain the same as in σ. In this case, end dates are not modified and we have:

- $end^\pi(a^p) = end^\sigma(a^p)$, $start^\pi(b^p) = start^\sigma(b^p)$
- $end^\pi(a^q) = end^\sigma(a^q)$, $start^\pi(b^q) = start^\sigma(b^q)$

If any of the conditions 12 and 13 is met, then it means that σ is not be precedence-feasible. This leads to a contradiction.

Case 2. $start^\sigma(a^p) > start^\sigma(a^q)$ and $start^\sigma(b^p) \leq start^\sigma(b^q)$ (Figure 9b). This means that in the schedule π, the start dates of a^p and a^q are swapped, but b^p and b^q remain the same as in σ. In this case, we have:

- $end^\pi(a^p) = end^\sigma(a^q)$; $start^\pi(b^p) = start^\sigma(b^p)$
- $end^\pi(a^q) = end^\sigma(a^p)$; $start^\pi(b^q) = start^\sigma(b^q)$

If Cond. 12 is true, then we have $end^\sigma(a^q) > start^\sigma(b^p)$. Since $start^\sigma(a^p) > start^\sigma(a^q)$ from hypothesis, then $end^\sigma(a^p) > end^\sigma(a^q)$ by the Definition 3. Therefore, we have $end^\sigma(a^p) > end^\sigma(a^q) > start^\sigma(b^p)$. As a result, σ is not precedence-feasible, leading to a contradiction.

If Cond. 13 is true, then we have $end^\sigma(a^p) > start^\sigma(b^q)$. Since $start^\sigma(b^q) \geq start^\sigma(b^p)$ from hypothesis, then we have $end^\sigma(a^p) > start^\sigma(b^p)$. As a result, σ is not precedence-feasible, leading to a contradiction.

Case 3. $start^\sigma(a^p) \leq start^\sigma(a^q)$ and $start^\sigma(b^p) > start^\sigma(b^q)$ (Figure 9c). This means that in the schedule π, the positions of a^p and a^q remain the same as in σ, but b^p and b^q are swapped. In this case, we have:

- $end^\pi(a^p) = end^\sigma(a^p)$; $start^\pi(b^p) = start^\sigma(b^q)$
- $end^\pi(a^q) = end^\sigma(a^q)$; $start^\pi(b^q) = start^\sigma(b^p)$

If Cond. 12 is true, then we have $end^\sigma(a^p) > start^\sigma(b^q)$. Since $start^\sigma(a^q) \geq start^\sigma(a^p)$ from hypothesis, then $end^\sigma(a^q) \geq end^\sigma(a^p)$ by the Definition 3. Therefore, we have $end^\sigma(a^q) \geq end^\sigma(a^p) > start^\sigma(b^q)$. As a result, σ is not precedence-feasible, leading to a contradiction.

If Cond. 13 is true, then we have $end^\sigma(a^q) > start^\sigma(b^p)$. Since we have $start^\sigma(b^p) > start^\sigma(b^q)$, then $end^\sigma(a^q) > start^\sigma(b^p) > start^\sigma(b^q)$. As a result, σ is not precedence-feasible, leading to a contradiction.

Case 4. $start^\sigma(a^p) > start^\sigma(a^q)$ and $start^\sigma(b^p) > start^\sigma(b^q)$ (Figure 9d). This means that in the schedule π, the positions between a^p and a^q, and between b^p and b^q are swapped. In this case, we have:

- $end^\pi(a^p) = end^\sigma(a^q)$; $start^\pi(b^p) = start^\sigma(b^q)$
- $end^\pi(a^q) = end^\sigma(a^p)$; $start^\pi(b^q) = start^\sigma(b^p)$

If any of the conditions 12 and 13 is met, σ will not be precedence-feasible, leading to a contradiction.

Every possible cases are leading to a contradiction, so π is precedence-feasible and lemma 6 is proved. ◀

C Lemma 7

Complete proof. We first define several elements for the proof.

Let $T^{\sigma}_{p,a}$ denote the time interval during which the activity $a \in \mathcal{A}_c$ of project p of a class c is active in a schedule σ, *i.e.* $T^{\sigma}_{p,a} = \{t \in [\![0, H]\!] | start^{\sigma}(a^p) \leq t < end^{\sigma}(a^p)\}$.

We then define a function representing the resource consumption of an activity within a project at a given time step in a schedule.

For any resource $r \in \mathcal{R}$, any time step $t \in [\![0, H]\!]$, any class $c \in \mathcal{C}$ and any activity $a \in \mathcal{A}_c$, we define a function γ that takes in parameter a schedule σ and a project $p \in \mathcal{I}_c$ and that has the following value:

$$\gamma_{r,a,t}(p, \sigma) = \begin{cases} cons_{r,a} & \text{if } t \in T^{\sigma}_{p,a} \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

The consumption of a resource r at each time step t in a schedule σ is given by the formula $\sum_{c \in \mathcal{C}, p \in \mathcal{I}_c, a \in \mathcal{A}_c} \gamma_{r,a,t}(p, \sigma)$.

Then, we define a function $\Delta_{r,a,t}(p, \sigma, \pi)$ representing the difference in resource consumption of resource r by activity a in project p between two schedules σ and π, at time step t. Formally,

$$\Delta_{r,a,t}(p, \sigma, \pi) = \gamma_{r,a,t}(p, \pi) - \gamma_{r,a,t}(p, \sigma)$$

We now consider a solution σ, a class $c \in \mathcal{C}$ and two projects $p, q \in \mathcal{I}_c$. We suppose the (p, q)-permutation π of σ is not resource-feasible. This means there exists a time moment $t \in [\![0, H]\!]$ and a resource $r \in \mathcal{R}$ such that $\sum_{d \in \mathcal{C}, u \in \mathcal{I}_d, a \in \mathcal{A}_d} \gamma_{r,a,t}(u, \pi) > capa_r$. Since σ is resource-feasible, we have $\sum_{d \in \mathcal{C}, u \in \mathcal{I}_d, a \in \mathcal{A}_d} \gamma_{r,a,t}(u, \sigma) \leq capa_r$. This implies:

$$\sum_{d \in \mathcal{C}, u \in \mathcal{I}_d, a \in \mathcal{A}_d} \big(\gamma_{r,a,t}(u, \pi) - \gamma_{r,a,t}(u, \sigma)\big) > 0$$

and then:

$$\sum_{d \in \mathcal{C}, u \in \mathcal{I}_d, a \in \mathcal{A}_d} \Delta_{r,a,t}(u, \sigma, \pi) > 0 \tag{15}$$

Following the definition of π (Definition 5), any class $d \in \mathcal{C}$, any project $u \in \mathcal{I}_d$ such that $u \neq p$ and $u \neq q$, and any activity $a \in \mathcal{A}_d$, $start^{\pi}(a^u) = start^{\sigma}(a^u)$ and $end^{\pi}(a^u) = end^{\sigma}(a^u)$. This means that for any time moment $t \in [\![0, H]\!]$, $\gamma_{r,a,t}(u, \sigma) = \gamma_{r,a,t}(u, \pi)$ and then $\Delta_{r,a,t}(u, \sigma, \pi) = 0$, i.e. the consumption of resource r by activity a^u does not change. Equation 15 can be simplified by only considering activities of projects p and q as:

$$\sum_{a \in \mathcal{A}_c} \big(\Delta_{r,a,t}(p, \sigma, \pi) + \Delta_{r,a,t}(q, \sigma, \pi)\big) > 0 \tag{16}$$

Case 1. Consider an arbitrary atomic activity $a \in \mathcal{A}^A_c$. The two virtual activities α_c and ω_c are not considered in the following as they do not consume resources (see Assumption 3). The two activities a^p and a^q are positioned in π according to the condition in Definition 5, that is:

$$start^{\sigma}(a^p) \leq start^{\sigma}(a^q) \implies start^{\pi}(a^p) = start^{\sigma}(a^p) \text{ and } start^{\pi}(a^q) = start^{\sigma}(a^q)$$
$$start^{\sigma}(a^p) > start^{\sigma}(a^q) \implies start^{\pi}(a^p) = start^{\sigma}(a^q) \text{ and } start^{\pi}(a^q) = start^{\sigma}(a^p)$$

In this case, we have $T^{\pi}_{p,a} \cup T^{\pi}_{q,a} = T^{\sigma}_{p,a} \cup T^{\sigma}_{q,a}$ and $T^{\pi}_{p,a} \cap T^{\pi}_{q,a} = T^{\sigma}_{p,a} \cap T^{\sigma}_{q,a}$ because:

- If $start^{\sigma}(a^p) \leq start^{\sigma}(a^q)$ then $T^{\pi}_{p,a} = T^{\sigma}_{p,a}$ and $T^{\pi}_{q,a} = T^{\sigma}_{q,a}$
- If $start^{\sigma}(a^p) > start^{\sigma}(a^q)$ then $T^{\pi}_{p,a} = T^{\sigma}_{q,a}$ and $T^{\pi}_{q,a} = T^{\sigma}_{p,a}$

Case 1.1. If there is no swap between a^p and a^q, then $T^{\pi}_{p,a} = T^{\sigma}_{p,a}$ and $T^{\pi}_{q,a} = T^{\sigma}_{q,a}$. Consumption of resource r is not modified at time t and we have $\Delta_{r,a,t}(p,\sigma,\pi) = \Delta_{r,a,t}(q,\sigma,\pi) = 0$.

Case 1.2. If there is a swap between a^p and a^q, then $T^{\pi}_{p,a} = T^{\sigma}_{q,a}$ and $T^{\pi}_{q,a} = T^{\sigma}_{p,a}$.

1. If $t \notin T^{\sigma}_{p,a} \cup T^{\sigma}_{q,a}$, then $t \notin T^{\pi}_{p,a} \cup T^{\pi}_{q,a}$. Both a^p and a^q are not being executed at time t in π, so

   $$\gamma_{r,a,t}(p,\sigma) = \gamma_{r,a,t}(q,\sigma) = \gamma_{r,a,t}(p,\pi) = \gamma_{r,a,t}(q,\pi) = 0$$

   This implies $\Delta_{r,a,t}(p,\sigma,\pi) = \Delta_{r,a,t}(q,\sigma,\pi) = 0$.
2. If $t \in (T^{\sigma}_{p,a} \cup T^{\sigma}_{q,a}) \setminus (T^{\sigma}_{p,a} \cap T^{\sigma}_{q,a})$, then $t \in (T^{\pi}_{p,a} \cup T^{\pi}_{q,a}) \setminus (T^{\pi}_{p,a} \cap T^{\pi}_{q,a})$. In both schedules σ and π, either a^p or a^q is being executed at time t.
   a. If $t \in T^{\sigma}_{p,a}$, $t \notin T^{\sigma}_{q,a}$, then we have $t \in T^{\pi}_{q,a}$ and $t \notin T^{\pi}_{p,a}$. This implies:

      $$\gamma_{r,a,t}(p,\sigma) = \gamma_{r,a,t}(q,\pi) = cons_{r,a} \; ; \; \gamma_{r,a,t}(q,\sigma) = \gamma_{r,a,t}(p,\pi) = 0$$

      and then:

      $$\Delta_{r,a,t}(p,\sigma,\pi) + \Delta_{r,a,t}(q,\sigma,\pi) = -cons_{r,a} + cons_{r,a} = 0$$

   b. Similarly, if $t \notin T^{\sigma}_{p,a}$, $t \in T^{\sigma}_{q,a}$ then:

      $$\gamma_{r,a,t}(p,\sigma) = \gamma_{r,a,t}(q,\pi) = 0 \; ; \; \gamma_{r,a,t}(q,\sigma) = \gamma_{r,a,t}(p,\pi) = cons_{r,a}$$

      This implies:

      $$\Delta_{r,a,t}(p,\sigma,\pi) + \Delta_{r,a,t}(q,\sigma,\pi) = cons_{r,a} - cons_{r,a} = 0$$
3. If $t \in T^{\sigma}_{p,a} \cap T^{\sigma}_{q,a}$, then $t \in T^{\pi}_{p,a} \cap T^{\pi}_{q,a}$. Both a^p and a^q are being executed at time t in π, so

   $$\gamma_{r,a,t}(p,\sigma) = \gamma_{r,a,t}(q,\sigma) = \gamma_{r,a,t}(p,\pi) = \gamma_{r,a,t}(q,\pi) = cons_{r,a}$$

   This implies $\Delta_{r,a,t}(p,\sigma,\pi) = \Delta_{r,a,t}(q,\sigma,\pi) = 0$.

From the above, we can conclude that the swap between two atomic activities never violates the resource constraint as $\Delta_{r,a,t}(p,\sigma,\pi) + \Delta_{r,a,t}(q,\sigma,\pi) = 0$. Therefore, Equation 16 can be rewritten as:

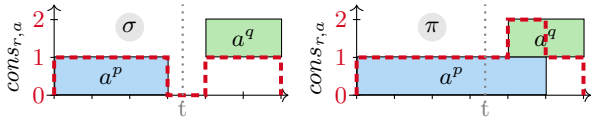
$$\sum_{a \in \mathcal{A}^C_c} \Delta_{r,a,t}(p,\sigma,\pi) + \Delta_{r,a,t}(q,\sigma,\pi) > 0 \tag{17}$$

Case 2. Consider an arbitrary compound activity $a \in \mathcal{A}^C_c$. There are two ways for making $\Delta_{r,a,t}(p,\sigma,\pi) + \Delta_{r,a,t}(q,\sigma,\pi)$ strictly positive.

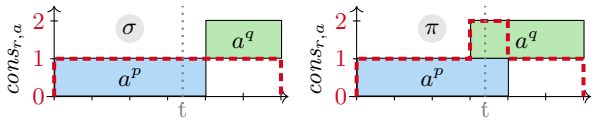
1. If activities a^p and a^q do not use r at time t in σ, but at least one of them uses it in π, then we have $\gamma_{r,a,t}(p,\sigma) = \gamma_{r,a,t}(q,\sigma) = 0$ and $\gamma_{r,a,t}(p,\pi) + \gamma_{r,a,t}(q,\pi) \geq cons_{r,a}$. This is expressed through the following condition.

   $$(t \notin T^{\sigma}_{p,a} \cup T^{\sigma}_{q,a}) \wedge (t \in T^{\pi}_{p,a} \cup T^{\pi}_{q,a}) \tag{18}$$
2. if exactly one activity (a^p or a^q) uses r at time t in σ and they both use it in π, then $\gamma_{r,a,t}(p,\sigma) + \gamma_{r,a,t}(q,\sigma) = cons_{r,a}$ and $\gamma_{r,a,t}(p,\pi) + \gamma_{r,a,t}(q,\pi) = 2cons_{r,a}$. This is expressed by

   $$(t \in (T^{\sigma}_{p,a} \cup T^{\sigma}_{q,a}) \setminus (T^{\sigma}_{p,a} \cap T^{\sigma}_{q,a})) \wedge (t \in T^{\pi}_{p,a} \cap T^{\pi}_{q,a}) \tag{19}$$



(a) Usage of resource r at time t: from 0 to $cons_{r,a}$.



(b) Usage of resource r at time t: from $cons_{r,a}$ to $2cons_{r,a}$.

Figure 10 Two impossible cases considered in proof of Lemma 7 when $a \in \mathcal{A}_c^C$.

Case 2.1. Let the condition 18 be true.

1. If $t < start^\sigma(a^p)$ and $t < start^\sigma(a^q)$ then for all $b \in \mathcal{A}_c$ with $(a,b) \in \mathcal{H}_c$, we have $t < start^\sigma(b^p)$ and $t < start^\sigma(b^q)$. We then have $t < start^\pi(b^p)$ and $t < start^\pi(b^q)$ and this implies $t \notin T^\pi_{p,a} \cup T^\pi_{q,a}$, leading to a contradiction.
2. If $t \geq end^\sigma(a^p)$ and $t \geq end^\sigma(a^q)$ then for all $b \in \mathcal{A}_c$ such that $(a,b) \in \mathcal{H}_c$, we have $t \geq end^\sigma(b^p)$ and $t \geq end^\sigma(b^q)$. We then have $t \geq end^\pi(b^p)$ and $t \geq end^\pi(b^q)$ and this implies $t \notin T^\pi_{p,a} \cup T^\pi_{q,a}$, leading to a contradiction.
3. If $end^\sigma(a^p) \leq t < start^\sigma(a^q)$ then for all $b \in \mathcal{A}_c$ such that $(a,b) \in \mathcal{H}_c$, we have $start^\sigma(b^p) \leq end^\sigma(b^p) \leq t < start^\sigma(b^q)$. This implies that zero swap operations were performed. Thus, $t \notin T^\pi_{p,a} \cup T^\pi_{q,a}$, leading to a contradiction.
4. If $end^\sigma(a^q) \leq t < start^\sigma(a^p)$ then for all $b \in \mathcal{A}_c$ such that $(a,b) \in \mathcal{H}_c$, we have $start^\sigma(b^q) \leq end^\sigma(b^q) \leq t < start^\sigma(b^p)$. We then have $start^\pi(b^p) \leq end^\pi(b^p) \leq t < start^\pi(b^q)$ and this implies $t \notin T^\pi_{p,a} \cup T^\pi_{q,a}$, leading to a contradiction.

All possible situations lead to a contradiction, so condition 18 is never satisfied.

Case 2.2. Let the condition 19 be true.

1. If $T^\sigma_{p,a} \cap T^\sigma_{q,a} = \emptyset$, then for all $t' \in T^\sigma_{p,a}$, either $t' < start^\sigma(a^q)$ or $t' \geq end^\sigma(a^q)$. In the first case, none activities from p and q are swapped, meaning that $T^\pi_{p,a} = T^\sigma_{p,a}$ and $T^\pi_{q,a} = T^\sigma_{q,a}$. In the second case, all activities from p and q are swapped, meaning that $T^\pi_{p,a} = T^\sigma_{q,a}$ and $T^\pi_{q,a} = T^\sigma_{p,a}$. In both cases, $T^\pi_{p,a} \cap T^\pi_{q,a} = T^\sigma_{p,a} \cap T^\sigma_{q,a} = \emptyset$, leading to a contradiction.
2. If $T^\sigma_{p,a} \cap T^\sigma_{q,a} \neq \emptyset$, there are four possible situations (see Figure 11):
   a. We suppose that $start^\sigma(a^p) \leq t < start^\sigma(a^q)$. Because π is a (p,q)-permutation of σ, for all activity $b \in \mathcal{A}_c$ such that $(a,b) \in \mathcal{H}_c$, we have $start^\sigma(b^q) \leq start^\pi(b^q)$. In particular, for each b such that $(a,b) \in \mathcal{H}_c$, we have $t < start^\sigma(b^q) \leq start^\pi(b^q)$. As $start^\pi(a^q)$ is the minimum value of start dates of b^q, we have $t < start^\pi(a^q)$. This implies that $t \notin T^\pi_{q,a}$, which contradicts Condition 19.
   b. We suppose that $start^\sigma(a^q) \leq t < start^\sigma(a^p)$. Let b be a child activity of a. Because $start^\sigma(a^p) \leq start^\sigma(b^p)$, we have $t < start^\sigma(b^p)$.
      - Suppose that $start^\sigma(b^q) < start^\sigma(a^p)$. As $start^\sigma(a^p) \leq start^\sigma(b^p)$, we have $start^\sigma(b^q) < start^\sigma(b^p)$. Therefore, b^q and b^p are swapped in π, which means that $start^\pi(b^q) = start^\sigma(b^p)$. We therefore have $t < start^\pi(b^q)$.
      - Suppose that $start^\sigma(b^q) \geq start^\sigma(a^p)$. Because start dates of activities of q in π are always greater or equal to those in σ, we have $start^\pi(b^q) \geq start^\sigma(b^q) \geq start^\sigma(a^p)$. This means that $t < start^\pi(b^q)$.

      We have $t < start^\pi(b^q)$ for all $b \in \mathcal{A}_c$ such that $(a,b) \in \mathcal{H}_c$, so $t < start^\pi(a^q)$ and then $t \notin T^\pi_{q,a}$, leading to a contradiction.

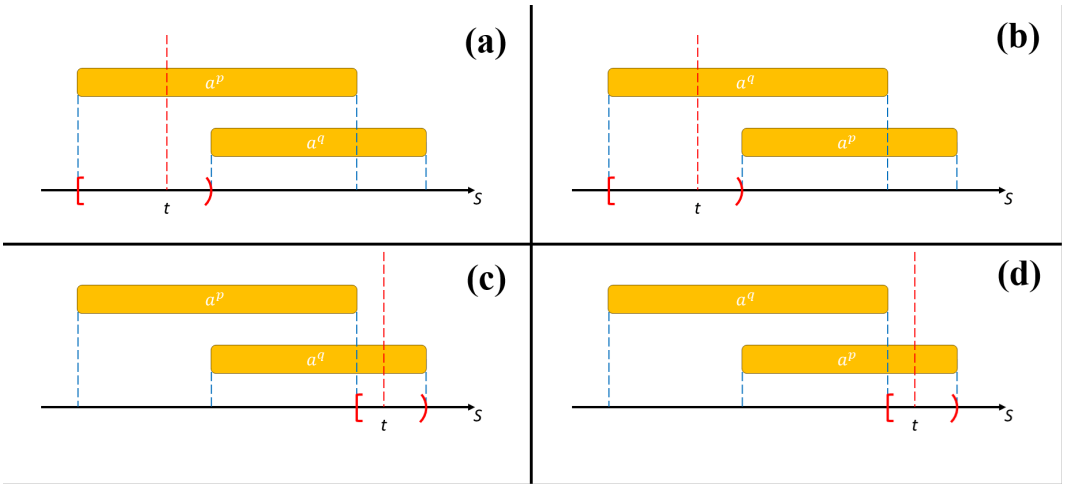


Figure 11 Four possible situation if $T^{\sigma}_{p,a} \cap T^{\sigma}_{q,a} \neq \emptyset$ (Case 2.2.2 – Lemma 7).

c. Let suppose that $end^{\sigma}(a^p) \leq t < end^{\sigma}(a^q)$. Let b be a child activity of a. Because $end^{\sigma}(b^p) \leq end^{\sigma}(a^p)$, we have $end^{\sigma}(b^p) \leq t$. π is a (p,q)-permutation so $end^{\pi}(b^p) \leq end^{\sigma}(b^p)$. So, $end^{\pi}(b^p) \leq t$.
This means that $end^{\pi}(a^p) \leq t$, which implies that $t \notin T^{\pi}_{p,a}$, leading to a contradiction.

d. Let suppose that $end^{\sigma}(a^q) \leq t < end^{\sigma}(a^p)$. Let b be a child activity of a. Because $end^{\sigma}(b^q) \leq end^{\sigma}(a^q)$, we have $end^{\sigma}(b^q) \leq t$.
- Let suppose that $end^{\sigma}(a^q) < end^{\sigma}(b^p)$. Then, $end^{\sigma}(b^q) < end^{\sigma}(b^p)$. With definition 3, this means that $start^{\sigma}(b^q) < start^{\sigma}(b^p)$. This condition implies that b^q and b^p are swapped in π. Therefore, $end^{\pi}(b^p) = end^{\sigma}(b^q)$. So $end^{\pi}(b^p) \leq t$.
- Now suppose that $end^{\sigma}(a^q) \geq end^{\sigma}(b^p)$. We have $end^{\sigma}(b^p) \geq end^{\pi}(b^p)$ (end dates of activities of p in σ are greater or equal to those in π). Therefore, $t \geq end^{\pi}(b^p)$.

We have $t \geq end^{\pi}(b^p)$ for all child b of a so $t \geq end^{\pi}(a^p)$. This means that $t \notin T^{\pi}_{p,a}$, leading to a contradiction.

All possible situations lead to a contradiction, so condition 19 is never satisfied. Both conditions 18 and 19 are never satisfied, so the sum $\Delta_{r,a,t}(p,\sigma,\pi) + \Delta_{r,a,t}(q,\sigma,\pi)$ is never positive.

In all cases, we have $\sum_{a \in \mathcal{A}^C_c} \Delta_{r,a,t}(p,\sigma,\pi) + \Delta_{r,a,t}(q,\sigma,\pi) \leq 0$ so the (p,q)-permutation π of σ is resource-feasible. ◀

D Lemma 11

Complete Proof. As all compound activities start and conclude with at least one atomic activity among their children, the end date of a project only depends on the end date of its atomic activities. Since σ is start-end-consistent, π is also start-end-consistent because the swap operation does not change the duration of atomic activities. Thus, we have $start^{\pi}(a^p) \leq start^{\pi}(a^q)$ and $end^{\pi}(a^p) \leq end^{\pi}(a^q), \forall a \in \mathcal{A}^A_c$. Because $p \prec q$, then for each activity $a \in \mathcal{A}^A_c$ that is swapped, we have $end^{\pi}(a^p) = end^{\sigma}(a^q) < end^{\sigma}(a^p)$. This implies that the statement $\tau^{\pi}_p \leq \tau^{\sigma}_p$ is always true.
Let $a, b \in \mathcal{A}^A_c$ such that a^p concludes project p and b^q concludes project q in the schedule σ. There are four possible situations:

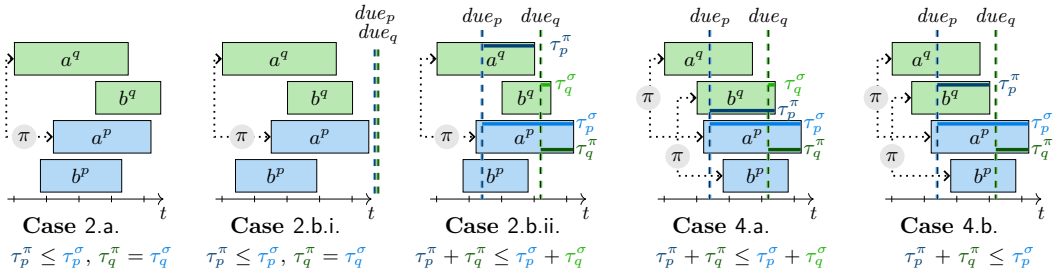


Figure 12 Illustration of cases considered in Lemma 11's proof.

1. Both a and b are not swapped in π then for each activity $i \in \mathcal{A}_c^A$, we have $end^\sigma(i^p) \leq end^\sigma(a^p) \leq end^\sigma(a^q) \leq end^\sigma(b^q)$ and $end^\sigma(i^q) \leq end^\sigma(b^q)$. Whether activity i is swapped or not, we always have $end^\pi(i^q) \leq end^\sigma(b^q) = end^\pi(b^q)$, so b^q is still the activity that concludes q in π and $\tau_q^\pi = \tau_q^\sigma$. Since $\tau_p^\pi \leq \tau_p^\sigma$ is always true, π is at least as good as σ in this criterion.
2. a is swapped but b is not swapped in π.
   a. If $end^\sigma(a^p) \leq end^\sigma(b^q)$ then for each activity $i \in \mathcal{A}_c^A$, we have $end^\sigma(i^p) \leq end^\sigma(a^p) \leq end^\sigma(b^q)$ and $end^\sigma(i^q) \leq end^\sigma(b^q)$. Whether activity i is swapped or not, we always have $end^\pi(i^q) \leq end^\sigma(b^q) = end^\pi(b^q)$, so b^q is still the activity that concludes q in π and $\tau_q^\pi = \tau_q^\sigma$. Since $\tau_p^\pi \leq \tau_p^\sigma$ is always true, π is at least as good as σ in this criterion.
   b. If $end^\sigma(a^p) > end^\sigma(b^q)$ then for each activity $i \in \mathcal{A}_c^A$, we have $end^\sigma(i^q) \leq end^\sigma(b^q) < end^\sigma(a^p)$ and $end^\sigma(i^p) \leq end^\sigma(a^p)$. Whether activity i is swapped or not, we always have $end^\pi(i^q) \leq end^\sigma(a^p) = end^\pi(a^q)$, so a^q is the activity that concludes q in π.
      i. If $end^\sigma(a^p) \leq due_q$ then $end^\pi(a^q) \leq due_q$. This implies $\tau_q^\pi = 0$ and since $\tau_p^\pi \leq \tau_p^\sigma$ is always true, π is at least as good as σ in this criterion.
      ii. If $end^\sigma(a^p) > due_q \geq due_p$, then $\tau_q^\pi = end^\sigma(a^p) - due_q$ and $\tau_p^\sigma = end^\sigma(a^p) - due_p$. For each activity $i \in \mathcal{A}_c^A$, there exists i^q such that $end^\sigma(i^q) \leq end^\sigma(b^q)$. Whether i is swapped or not in π, we always have $end^\pi(i^p) \leq end^\sigma(b^q)$. Thus we have $\tau_p^\pi \leq \max(0, end^\sigma(b^q) - due_p)$.
      We also have $\tau_q^\sigma = \max(0, end^\sigma(b^q) - due_q)$. We next consider three cases for the relative values for $end^\sigma(b^q)$, due_p and due_q.
         A. If $end^\sigma(b^q) \geq due_q$, then $end^\sigma(b^q) \geq due_p$. So $\tau_p^\pi \leq end^\sigma(b^q) - due_p$. We also have $\tau_q^\sigma = end^\sigma(b^q) - due_q$.
         Therefore, $\tau_p^\pi + \tau_q^\pi \leq end^\sigma(b^q) - due_p + end^\sigma(a^p) - due_q$. The right member of that inequality is exactly equal to $\tau_p^\sigma + \tau_q^\sigma$ so $\tau_p^\pi + \tau_q^\pi \leq \tau_p^\sigma + \tau_q^\sigma$.
         B. If $due_p \leq end^\sigma(b^q) \leq due_q$, then $\tau_q^\sigma = 0$ and $\tau_p^\pi \leq end^\sigma(b^q) - due_p$. Then, we have: $\tau_p^\pi + \tau_q^\pi \leq end^\sigma(b^q) - due_p + end^\sigma(a^p) - due_q$. The right part is equal to $end^\sigma(b^q) - due_q + \tau_p^\sigma$. Because $end^\sigma(b^q) \leq due_q$, we have $\tau_p^\pi + \tau_q^\pi \leq \tau_p^\sigma$. As $\tau_q^\sigma = 0$, we have $\tau_p^\pi + \tau_q^\pi \leq \tau_p^\sigma + \tau_q^\sigma$.
         C. If $end^\sigma(b^q) \leq due_p$, then $end^\sigma(b^q) \leq due_q$. This means that $\tau_p^\pi = \tau_q^\sigma = 0$. Therefore, $\tau_p^\pi + \tau_q^\pi = end^\sigma(a^p) - due_q$. Because $due_p \leq due_q$, $\tau_p^\pi + \tau_q^\pi \leq end^\sigma(a^p) - due_p$, where the left part is equal to τ_p^σ. As $\tau_q^\sigma = 0$, then $\tau_p^\pi + \tau_q^\pi \leq \tau_p^\sigma + \tau_q^\sigma$.
      We have shown that in this case, $\tau_p^\pi + \tau_q^\pi \leq \tau_p^\sigma + \tau_q^\sigma$ so π is at least as good as σ for the tardiness criterion.
3. a is not swapped but b is swapped in π. This means $end^\sigma(a^p) \leq end^\sigma(a^q)$ and $end^\sigma(b^q) < end^\sigma(b^p)$. As a^p concludes p in σ, we have $end^\sigma(b^q) < end^\sigma(b^p) \leq end^\sigma(a^p) \leq end^\sigma(a^q)$. This violates the condition that b^q concludes q in σ, so this situation never happen.

4. Both a and b are swapped in π. We have $end^{\sigma}(a^q) < end^{\sigma}(a^p)$ and $end^{\sigma}(b^q) < end^{\sigma}(b^p)$. Because a^p concludes p and b^q concludes q in σ, then:
   $end^{\sigma}(a^q) \leq end^{\sigma}(b^q) < end^{\sigma}(b^p) \leq end^{\sigma}(a^p)$
   For each activity $i \in \mathcal{A}_c^A$, i^q and i^p are such that $end^{\sigma}(i^q) \leq end^{\sigma}(b^q)$ and $end^{\sigma}(i^p) \leq end^{\sigma}(a^p)$. Whether i is swapped or not in π, we always have $end^{\pi}(i^p) \leq end^{\sigma}(b^q) = end^{\pi}(b^p)$ and $end^{\pi}(i^q) \leq end^{\sigma}(a^p) = end^{\pi}(a^q)$. Thus, we have b^p concludes p and a^q concludes q in π. If:
   a. q is late in σ then p is also late in σ because $due_p \leq due_q$ and $end^{\sigma}(b^q) < end^{\sigma}(a^p)$. We have:
      $due_p \leq due_q < end^{\sigma}(b^q) = end^{\pi}(b^p) < end^{\sigma}(a^p) = end^{\pi}(a^q)$
      So both p and q are late in π. We have:
      $\tau_p^{\pi} + \tau_q^{\pi} = end^{\pi}(b^p) - due_p + end^{\pi}(a^q) - due_q$
      $= end^{\sigma}(b^q) - due_p + end^{\sigma}(a^p) - due_q$
      $= \tau_p^{\sigma} + \tau_q^{\sigma}$
      So π and σ are equivalent in this criterion.
   b. q is early and p is late in σ then $\tau_p^{\sigma} = end^{\sigma}(a^p) - due_p$ and $\tau_q^{\sigma} = 0$. Since a^p is the last finished activity for both projects in σ and $due_p \leq due_q$, the value $end^{\sigma}(a^p) - due_p$ is the maximal possible tardiness of both projects in both schedules, *i.e.*, we have $\tau_p^{\pi} < end^{\sigma}(a^p) - due_p$ and $\tau_q^{\pi} < end^{\sigma}(a^p) - due_p$. If:
      i. At least one between p and q is early in π, then at lease one between τ_p^{π} and τ_q^{π} is equal to zero. Thus, $\tau_p^{\pi} + \tau_q^{\pi} < end^{\sigma}(a^p) - due_p = \tau_p^{\sigma} + \tau_q^{\sigma}$, so π is better than σ in this criterion.
      ii. Both p and q are late in π, then we have $\tau_p^{\pi} = end^{\pi}(b^p) - due_p = end^{\sigma}(b^q) - due_p$ and $\tau_q^{\pi} = end^{\pi}(a^q) - due_q = end^{\sigma}(a^p) - due_q$. We have:
      $\tau_p^{\pi} + \tau_q^{\pi} = end^{\sigma}(b^q) - due_p + end^{\sigma}(a^p) - due_q$
      $= \tau_p^{\sigma} + end^{\sigma}(b^q) - due_q$
      As q is early in σ then $end^{\sigma}(b^q) - due_q < 0$. This implies $\tau_p^{\pi} + \tau_q^{\pi} < \tau_p^{\sigma} = \tau_p^{\sigma} + \tau_q^{\sigma}$, so π is better than σ in this criterion.
   c. Both p and q are early in σ, then we have:
      $end^{\sigma}(b^q) < end^{\sigma}(a^p) \leq due_p \leq due_q$
      and then:
      $end^{\pi}(b^p) < end^{\pi}(a^q) \leq due_p \leq due_q$
      This implies that both p and q are early in π, so π and σ are equivalent in this criterion.

From the above, we can conclude that π is at least as good as σ in this criterion. ◀