Mixed-integer/linear and constraint programming approaches for activity scheduling in a nuclear research facility

Oliver Polo Mejia, Christian Artigues, Pierre Lopez, Virginie Basini

Mixed-integer/linear and constraint programming approaches for activity scheduling in a nuclear research facility

Oliver Polo-Mejía[a,b], Christian Artigues[a], Pierre Lopez[a], Virginie Basini[b]

[a]LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France ;
[b]CEA, DEN, DEC, SETC, Saint-Paul-lez-Durance, France

ARTICLE HISTORY
Compiled September 15, 2021

ABSTRACT
This paper presents the results of a research project aiming to optimise the scheduling of activities within a research laboratory of the 'Commissariat à l'Energie Atomique et aux Energies Alternatives (CEA)'. To tackle this problem, we decompose every activity into a set of elementary tasks to apply standard scheduling methods. We model the problem as an extended version of the Multi-Skill Project Scheduling Problem (MSPSP). As a first approach, we propose a Multi-Skill Project Scheduling Problem with penalty for preemption, along with its mixed-integer/linear programming (MILP) formulation, where the preemption is allowed applying a penalty every time an activity is interrupted. However, the previous approach does not take into account all safety constraints at the facility, and a more accurate variant of the problem is needed. We propose then to integrate the concept of partial preemption to the MSPSP. This concept, that has not been yet studied in the scientific literature, implies that only a subset of resources is released during preemption periods. The resulting MSPSP with partial preemption (MSPSP-PP) is modelled using two methodologies: MILP and constraint programming. Regarding the industrial need of having good solutions in a short time, we also presented a greedy algorithm for the MSPSP-PP.

KEYWORDS
Multi-skill research project; Scheduling; Partial preemption; Nuclear facility; Mixed-Integer/Linear Programming; Constraint Programming; Greedy heuristic

CONTACT Oliver Polo-Mejía. Email: oliver.polo-mejia@laas.fr

1. Introduction

The hot lab LECA-STAR is a nuclear research facility dedicated to R&D on irradiated fuels. It plays an essential role in the research projects of the Alternative Energies and Atomic Energy Commission or CEA (in French: Commissariat à l'Energie Atomique et aux Energies alternatives), since it ensures the execution of most of the post-irradiation experiences over nuclear fuel. Given the characteristics of the activities carried out at the LECA-STAR, an improvement on the scheduling process could be beneficial for ensuring the best performance of the facility.

Scheduling activities within a nuclear facility is a complex process because of the many operational and regulatory constraints to be taken into account in the nuclear field. For example, activities require for their execution the allocation of well-trained staff having particular skills and authorisations that not all members of the staff may have. This makes more complex the choice of the staff members that can execute each activity. The scheduling process becomes even harder when we talk about nuclear research facilities, since we must schedule activities that most of the time are not standardised. An improvement of the scheduling process of such facilities is a paramount issue.

A literature review allows identifying some applications of scheduling models within a nuclear environment, all of them for broad scheduling horizons. Chen et al. (2016), for example, proposed a heuristic method to solve the nuclear power plant construction scheduling problem, that integrates building construction scheduling and reactor installation scheduling. Petersen (2016) presents various methods aiming to schedule the removal of spent nuclear fuel from reactor sites in the USA. His objective was to reduce the amount of time the shutdown reactors keep the spent fuel on-site, and thus reduce the total system costs for the federal government. In France, *Electricité de France (EDF)*, the largest European producer of electricity, has used combinatorial optimisation techniques to schedule outages and maintenance of nuclear power plants (Dupin and Talbi 2016; Jost and Savourey 2013).

The activities carried out at the LECA-STAR are very close to a classical R&D project. Scheduling R&D projects is a complex process. This complexity lies mainly in the fact that the order and type of activities to be carried out during the project can vary greatly depending on the results obtained in the early stages. To handle the R&D project scheduling problem, Reyck and Leus (2008) have used different techniques to include uncertainties in the scheduling models such as robust optimisation (Hassanzadeh et al. 2014; Khemakhem and Chtourou 2013) or fuzzy optimisation (Gavareshki 2004; Norouzi et al. 2015; Pérez et al. 2018). In the same way that for nuclear-related scheduling activities, most of the literature of R&D project scheduling deals with broad scheduling horizons.

Working with a relatively short scheduling horizon, which is the case in this article (about a week), may reduce the subjectivity of the activities to be scheduled in a research project, thus allowing the use of an elementary activity approach, as proposed by Mancel (2004) for the scheduling of research activities for the Mars Netlander project. In her approach, each experiment consists of a series of basic tasks that are well defined (known duration, resource requirement, etc.). This elementary task approach allows us to use standard scheduling methods to schedule the activities of the LECA-STAR research laboratory involving a short scheduling horizon. Another pragmatic reason for discarding uncertain approaches is that the required data to describe activity uncertainties is not available. After analysing the characteristics of the activities carried out in the laboratory, we identify in this paper that the problem at

hand can be modelled as an extended version of an existing scheduling problem from the literature: the Multi-Skill Project Scheduling Problem (MSPSP) (Néron 2002), and, more precisely by two variants : a MSPSP with penalty for preemption and a MSPSP with partial preemption.

The contribution of the paper compared to previous work is as follows. We explain in this paper with satefety considerations why the second variant is a better model for the industrial case study. As the weekly scheduling problem involves about 100 activities, which in the scheduling literature is generally considered as moderately large, exact MILP and/or CP-based approaches are worth investigating. In previous work, a mixed-integer linear programming (MILP) formulation was proposed for the MSPSP with penalty for preemption in (Polo-Mejía et al. 2017). A MILP and a constraint programming (CP) formulation were proposed in (Polo-Mejía et al. 2018) for the MSPSP with partial preemption. In both cases, the non trivial industrial requirement that the same technicians must be present during the whole execution for non-preemptive activities was ignored. In this paper, we modify the MILP model for the MSPSP with penalty for preemption, presented in (Polo-Mejía et al. 2017), to incorporate this constraint. We also present new MILP and constraint programming (CP) formulations for the Multi-Skill Project Scheduling Problem with partial preemption that outperform the previous models when this constraint is considered. This allows us to solve exactly instances with larger sizes (30 activities vs 15 activities in (Polo-Mejía et al. 2018)). Additionally, to comply with the industrial need of fast computation times for larger instances, we describe a greedy algorithm for the MSPSP with partial preemption that also improves the heuristic previously proposed in (Polo-Mejía et al. 2019)) by incorporating priority rule based on the relaxation of the MILP models of the two variants.

The remainder of this paper is structured as follows: In the next section, we describe some important aspects of how the research facility works and that we must take into consideration during the modelling process. After this description, we present in Section 3 the first model we propose to represent the scheduling process of the laboratory, the MSPSP with penalty for preemption, and the associated MILP formulation. In Section 4, we present a more accurate variant, called Multi-Skill Project Scheduling Problem with partial preemption (MSPSP-PP). This variant uses the concept of partial preemption, which implies that only a subset of resources is released during the preemption periods. In the same section, we present the MILP and CP formulations for the proposed problem, as well as the greedy heuristic. Experimental results for the MSPSP-PP are presented in Section 5. Finally, we conclude in Section 6.

2. Industrial problem description

Every week more than 100 activities are carried out at LECA-STAR, including maintenance (preventive or curative), experimental activities, nuclear transport and regulatory controls. The laboratory runs its operations continuously from Monday morning to Friday night (108 hours). However, not all activities can be scheduled at any moment due to the absence (known in advance) of some resources and staff during specific periods. Due to this 'calendarisation', we can not guarantee the continuous execution of activities with a duration larger than the work shifts. Additionally, sometimes we must preempt (stop an activity in process to continue it later) non-critical activities (such as certain experimental activities) to give priority to more critical activities (such as nuclear transports that have stringent constraints for scheduling). Allowing

the preemption is then necessary for modelling our problem. A preempted activity can be resumed later by a set of resources different from the one used to start it. On the other hand, there is a subset of particular activities (set N of non-preemptive activities) that must be executed without interruption due to safety and operational constraints.

Activities carried out at LECA-STAR require the allocation of technicians (staffs) with particular skills and authorisations to be executed. In other words, not all technicians can execute all activities. Thus, activities are defined by their need for resources (equipment, machines, building) and their need for skills. Each technician has a specific set of skills it masters; we assume this mastering is done at the same level for each skill. His periods of presences/absences (calendarisation) are defined in advance and are not subject to change in this study. Technicians can be allocated to only one activity at a time, but they can execute several skills per activity at the same time. For example, the same technician can be responsible for recording the movement of nuclear material in the database (not everyone can do it), and at the same time, he performs a cut of the sample. However, for operational reasons, we must guarantee the allocation of a minimal number of technicians for an activity. For instance, activities in contaminated or isolated zones require at least two members of staff for surveillance.

The presence of time windows is also important in the nuclear facility. Nuclear regulation, for example, requires to carry out a series of periodic tests to ensure the proper functioning of the machines. These tests must be scheduled and executed before a deadline. Additionally, some of the activities are carried out in partnership with other laboratories, and they may be subject to receipt of a sample at a fixed contractual date, and the activity cannot start before such date. In this case, we say that the activity is subject to a release date. Sometimes an experimental or maintenance activity may require a set of setup activities. In this case, a precedence relationship exists between these activities (e.g. activity i cannot start before activity l is completed).

The scheduling of the activities for the following week must be constructed and validated during a meeting between the heads of research, heads of maintenance, and engineers responsible for activities that take place at the end of the week. Although the planners prepare an interim scheduling before this meeting, it is common that changes must be made during the meeting due to new information or the status of the administrative progress of the documentation necessary to carry out an activity. It is then crucial to be able to have a new feasible schedule within a few minutes. Hence, even if the moderate problem size allow to consider the usage of exact methods such as the MILP and CP based methods presented in this paper, heuristics are also a good alternative to exploit the rigorousness of combinatorial optimisation models while obtaining good quality (not necessarily optimal) schedules in a short time.

3. MSPSP with penalty for preemption

The characteristics of the scheduling process in the nuclear laboratory, where we must allocate a group of technicians mastering a specific set of skills to execute an activity, led us to conclude that the facility scheduling problem should be modelled as an extension of the Multi-Skill Project Scheduling Problem (MSPSP). This problem has been successfully used in various fields where skill requirements are critical for project development, such as R&D (Certa et al. 2009) and IT product development (Chen et al. 2017).

The MSPSP, presented for the first time by Néron (2002), is an extension of the

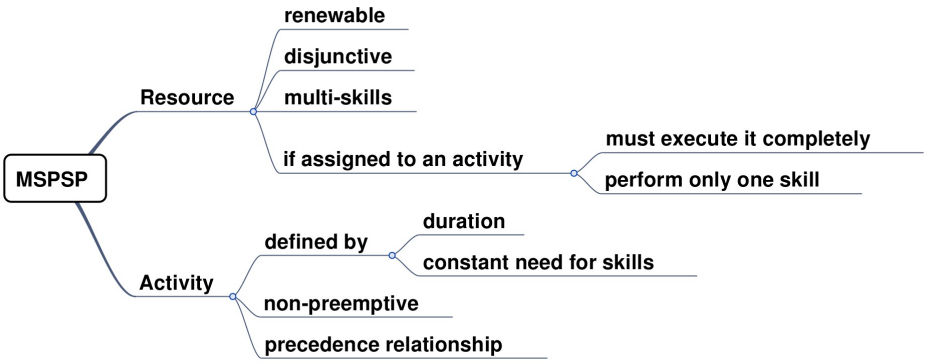

Figure 1. Characteristics of the Multi-Skill Project Scheduling Problem (MSPSP).

Resource-Constrained Project Scheduling Problem (RCPSP) (Artigues 2008), where resources are characterised by the skills they master, and tasks require a certain amount of resources with a specific skill. The objective is to find a feasible schedule minimising the project makespan while satisfying the precedence relationships and the resource constraints: a resource cannot execute a skill it does not master, cannot be assigned to more than one skill requirement at a given time, and must be assigned to the corresponding activity during its whole processing time (Bellenguez-Morineau 2008). The characteristics of the basic version of the problem are presented in Figure 1. Although the classic version of the MSPSP is very powerful, being able to model a large number of scheduling problems, we must make some modifications over the baseline version to get a most accurate representation of our industrial scheduling problem.

The most significant change is related to the possibility of interrupting an activity once it has started. The non-preemption constraints (activities execution must be without interruption) is one of the main characteristics of the MSPSP (Néron 2002). In our study, it is necessary to allow the preemption of a set of activities, while still forbidding this preemption for the remaining activities. Nuclear regulation requires that, for some critical activities, the workspace must be put in a safe configuration whenever an activity is interrupted, which means a loss of time and a decrease of the productivity. We must then try to limit the number of times these activities are preempted. Firstly, a suitable approach is to use an MSPSP with penalty for preemption where we apply a penalty (M_i) every time an activity i, for which we want to limit the preemption, is preempted. This penalty is then minimised in the objective function. With this in mind, we can classify the activities into three sets: non-preemptive activities (N), preemptive activities (P) and preemptive activities with penalty (W).

The second set of changes is related to the characteristics of the resources we model. The MSPSP, as defined by Néron (2002), works only with disjunctive resources, which means they have a unitary capacity. In our industrial variant, we must work with both disjunctive multi-skilled resources (technicians) and cumulative (i.e. they can handle more than one activity at a time) mono-skilled resources (compound machines and buildings). The technicians can execute several skills per activity at the same time. However, we must ensure a minimal number of technicians to comply with safety and operational constraints. Additionally, since some activities have a duration larger than technicians work shifts, the technicians can partially execute the activities, except

for non-preemptive activities, $i \in N$, which duration is smaller than work shifts and cannot be interrupted. Finally, basic formulations of the RCPSP and the MSPSP assume that resources are available in constant amounts during the whole planning horizon (Klein 2000). Since representing the absences of technicians over time is crucial in our problem, we are heading towards a modelling including time-varying resource capacity.

Traditional MSPSP does not take into account time windows for scheduling the activities. As stated in Section 2, some of the activities within the nuclear facility may be subject to time windows. These activities must be scheduled after a fixed release date (r_i) and/or before a deadline (dl_i). All other characteristics are similar to classical MSPSP. Now defined the characteristics of the problem, we can describe its mathematical formulation.

3.1. *Mixed-Integer/Linear Programming (MILP)*

Various MILP formulations, continuous and discrete-time based, have been proposed in the literature for the MSPSP. For the continuous-time based models, most of the time, authors have used precedence-based variables, as in (Correia et al. 2012) with additional inequalities aiming to enhance the model. A precedence-based non-linear model for the MSPSP is presented by Kazemipoor et al. (2013). The authors show later how the model can be linearised through a series of variable conversions. A MILP formulation, having similarities with the model proposed by Correia et al. (2012), was priorly proposed by Li and Womer (2009) for the MSPSP with minimal and maximal time lags. Discrete-time based formulations have been proposed by Bellenguez-Morineau (2006), Montoya et al. (2014) (Correia and Saldanha-da Gama (2015) proposed minor corrections to this model) and Almeida et al. (2019), all of them using binary time-indexed step variables. For a given activity i, the binary step variables $Z_{i,t}$ with t lower than the starting time of the activity are all equal to 0, whereas the variables with t equal to or greater than the starting time are all equal to 1.

MILP formulations for the preemptive MSPSP are very scant in the scientific literature. Moreover, the few ones we could find use time-indexed on/off variables. On/off variables take the value 1 for all the periods t where the activity is in execution. Maghsoudlou et al. (2018) proposed a MILP formulation for the preemptive MSPSP with due dates. They used an on/off variable $Z_{i,k,t}$, taking the value 1 if part k of activity i is executed at time t, together with another on/off variable ($X_{i,m,k,t}$) indicating the periods t at which each technician m works over each part k of activity i. An allocation binary variable ($Y_{i,m,k,s}$) is also used to indicate the skill s that every technician m performs over each part k of each activity i. Continuous variables are used to determine the earliness, tardiness and completion time of each activity. Dhib et al. (2015) also formulated a MILP for a preemptive variant of the MSPSP where all parts of an activity, each part corresponding to one skill requirement, must start simultaneously but can be preempted at different times. The authors used an on/off variable $X_{i,k,l,t}$ taking the value 1 if resource k performs the skill l for activity i during time t. Continuous auxiliary variables are used to calculate the starting and completion times of activities and the completion time of each skill for every activity. The on/off formulation has been identified as the most accurate for modelling the time-varying resource availability and activity preemption. Two different MILP models for the MSPSP with partial preemption are evaluated in Polo-Mejía et al. (2017); the one giving the best results is presented below:

Notation for parameters. All parameters used in the MILP model are shown in Table 1.

Variables. Variables used to model the MSPSP with penalty for preemption are given in Table 2.

Objective function. To assure the normal progress of research projects, the set of weekly activities must be executed in the shortest possible time. This can be translated as the minimisation of the project makespan:

$$\min(C_{\max})$$

In our case, we must add to this function a term minimising the penalties (M_i) due to the preemption of critical activities (those activities for which we want to limit the preemption). Our problem can then be modelled as:

$$\min\left(C_{\max} + \sum_{i \in W}\left(M_i * \left(\left(\sum_{t=r_i}^{dl_i} X_{i,t}\right) - 1\right)\right)\right) \tag{1}$$

s.t.

$$\sum_{i \in I}(Y_{i,t} * br_{i,k}) \leq DR_{k,t} \qquad \forall t \in H, \forall k \in K \tag{2}$$

$$\sum_{i \in I} O_{j,i,t} \leq DO_{j,t} \qquad \forall j \in J, \forall t \in H \tag{3}$$

$$Y_{i,t} * bs_{i,s} \leq \sum_{j \in J}(O_{j,i,t} * CO_{j,s}) \quad \forall i \in I, \forall t \in H, \forall s \in S \tag{4}$$

$$\sum_{j \in J} O_{j,i,t} \geq Y_{i,t} * nt_i \qquad \forall t \in H, \forall i \in I \tag{5}$$

$$\sum_{t=\max(1,r_i)}^{\min(dl_i,T)} Y_{i,t} \geq D_i \qquad \forall i \in I \tag{6}$$

$$D_i * (1 - Y_{l,t}) \geq \sum_{t'=t}^{T} Y_{i,t'} \qquad \forall (i,l) \in E, \forall t \in H \tag{7}$$

$$X_{i,t} \geq Y_{i,t} - Y_{i,t-1} \qquad \forall i \notin P, \forall t \geq 2 \tag{8}$$

$$X_{i,1} = Y_{i,1} \qquad \forall i \notin P \tag{9}$$

$$\sum_{t=1}^{T} X_{i,t} \leq 1 \qquad \forall i \in N \tag{10}$$

$$O_{j,i,t} \geq S_{j,i} + Y_{i,t} - 1 \qquad \forall j \in J, \forall t \in H, \forall i \in N \tag{11}$$

$$O_{j,i,t} \leq S_{j,i} \qquad \forall j \in J, \forall t \in H, \forall i \in N \tag{12}$$

$$C_{\max} \geq t * Y_{i,t} \qquad \forall i \in I, \forall t \in H \tag{13}$$

$$Y_{i,t},\ O_{j,i,t},\ S_{j,i},\ X_{i,t} \in \{0,1\}$$

Table 1. Parameters for the MSPSP with penalty for preemption.

Notations for parameters	
I	Set of activities to be scheduled
$N \subseteq I$	Set of non-preemptive activities
$P \subseteq I$	Set of preemptive activities
$W \subseteq I$	Set of preemptive activities with penalty
$E(i,l)$	Set of precedence relationships. State that activity $l \in I$ cannot start before activity $i \in I$ is completed
K	Set of mono-skilled resources
S	Set of skills
J	Set of available technicians
T	Last period of the scheduling horizon
$H = \{1, 2, ..., T\}$	Set of scheduling periods
$DR_{k,t} \in \mathbb{Z}_+$	Amount of resource $k \in K$ available at time $t \in H$
$br_{i,k} \in \mathbb{Z}_+$	Amount of resource $k \in K$ required for executing activity $i \in I$
$DO_{j,t} \in \{0,1\}$	Presence/absence of technician $j \in J$ at time $t \in H$. Equal to 1 if present, 0 otherwise
$CO_{j,s} \in \{0,1\}$	Indicates whether a technician $j \in J$ masters skill $s \in S$ or not. Equal to 1 if mastered, 0 otherwise
$bs_{i,s} \in \mathbb{Z}_+$	Amount of technicians mastering skill $s \in S$ required for executing activity $i \in I$
$nt_i \in \mathbb{Z}_+$	Minimal number of technicians required to execute activity $i \in I$
$D_i \in \mathbb{Z}_+$	Duration of activity $i \in I$
$r_i \in H$	Release date for activity $i \in I$
$dl_i \in H$	Deadline for activity $i \in I$
$M_i \in \mathbb{Z}_+$	Penalty for preempting activity $i \in W$ (expressed in time units)

Table 2. Variables for the MSPSP with penalty for preemption.

Notation for variables	
$Y_{i,t} \in \{0,1\}$	$Y_{i,t} = 1 \iff$ activity $i \in I$ is executed during time $t \in H$
$O_{j,i,t} \in \{0,1\}$	$O_{j,i,t} = 1 \iff$ technician $j \in J$ is allocated to activity $i \in I$ during time $t \in H$
$S_{j,i} \in \{0,1\}$	$S_{j,i} = 1 \iff$ technician $j \in J$ is assigned to execute activity $i \in N$
$X_{i,t} \in \{0,1\}$	$X_{i,t} = 1 \iff$ a part of activity $i \notin P$ starts at time $t \in H$
$C_{\max} \in \mathbb{R}_+$	Project makespan

$$C_{\max} \in \mathbb{R}_+$$

Constraints (2) guarantee that the total demand of each resource during all periods $t \in H$ is always lower than the resource capacity. Constraints (3) ensure that technicians are allocated in period t only if they are available. They also ensure that the technicians can be allocated to at most one activity during period t. Skill requirements and a minimal number of technicians are ensured by Constraints (4) and (5), respectively. Constraints (6) ensure that the durations of the activities are satisfied; these constraints also limit the activity executions between their release dates (r_i) and deadlines (dl_i) when there exist. Constraints (7) represent precedence relationships. The start time of each part of an activity is determined by Constraints (8) and (9). With inequalities (10) we ensure that preemption is not allowed for non-preemptive activities. Constraints (11) and (12) ensure that if a technician is allocated to a non-preemptive activity it must execute it until completeness. These constraint were not present in the model presented in Polo-Mejía et al. (2017). Finally, the makespan of the project is calculated using constraints (13).

Even if the MSPSP with penalty for preemption allows to model an important number of the activities carried out at the LECA-STAR, it does not fulfil all the safety constraints for a subset of activities. That is why in the next section, we present a more accurate model that takes into account these additional safety constraints.

4. MSPSP with partial preemption (MSPSP-PP)

Typically, preemptive scheduling problems assume that all resources are released during preemption periods, and that they can be used to perform other activities. Nevertheless, at the LECA-STAR, safety constraints require that a subset of resources remains allocated to the activity when it has been preempted. Suppose one must execute an experimental activity that requires an inert atmosphere for its execution. In practice, one can stop this activity and allow the technicians and some of the equipment to be used in other activities. However, safety and operational constraints force us to preserve the inert atmosphere even when the activity is stopped until it ends. In other words, one cannot release the equipment that ensures the inert atmosphere during the preemption periods. Traditional preemptive schedule models cannot represent this behaviour. Until now, the only way to model this activity, while respecting safety requirements, was to declare it as 'non-preemptive'. However, this decision can increase the project makespan, especially in our case-study, where the activities may have restrictive time-windows and the availability/capacity of the resources varies over time. The possibility of only releasing a subset of resources during the preemption periods, what we call *partial preemption*, has not yet been studied in the scientific literature. To comply with this safety requirement, and to fill the gap in the literature, we present a new variant of the MSPSP that uses the concept of partial preemption: the MSPSP with partial preemption or MSPSP-PP.

In the MSPSP-PP, if an activity is preempted, we release only a subset of resources while seizing the others. We can then classify the activities in three types according to the possibility of releasing the resources during the preemption periods: 1) Non-preemptive activities (N), if none of the resources can be released; 2) Partially preemptive activities (PP), if a subset of resources can be released; and 3) Preemptive activities (P), if all resources can be set free. In our case, the partial preemption is

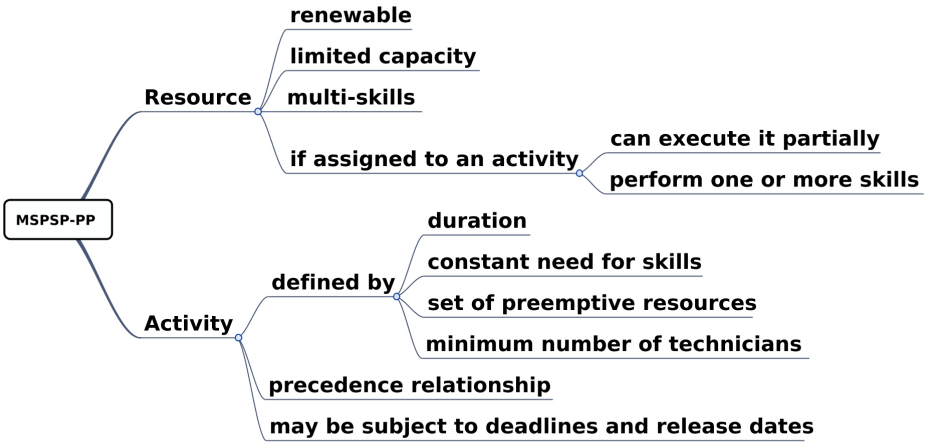

Figure 2. Characteristics of the Multi-Skill Project Scheduling Problem with Partial Preemption (MSPSP-PP).

only related to mono-skilled resources, and we made the hypothesis that technicians can always be released during preemption periods. This is because, in practice, we are not interested in allocating staff to an activity that is not in execution. All other characteristics are the same as those presented in Section 3 and are summarised in Figure 2.

Summarising, the objective in the MSPSP with partial preemption is to find a feasible schedule that minimises the total duration of the project ($C_{\max}$). Finding a solution consists in determining the periods during which each activity is executed and also which resources will execute the activity in every period, while satisfying all the constraints. We must schedule these activities over renewable resources with limited capacity; they can be cumulative mono-skilled resources (machines or equipment) or disjunctive multi-skilled resources (technicians) mastering Nb_j skills. Multi-skilled resources can respond to more than one skill requirement per activity and may execute it partially (except for non-preemptive activities where technicians must perform the whole activity). An activity is defined by its duration (D_i), its precedence relationships, its requirements of resources ($br_{i,k}$), its requirements of skills ($bs_{i,s}$), the minimum number of technicians needed to perform it (nt_i) and the subset of preemptive resources. Activities might or not have either a deadline (dl_i) or a release date (r_i). Figure 3 illustrates an example of an MSPSP-PP instance and a possible solution.

Using the concept of partial preemption, we can model not only the non-preemption constraints linked to safety but also we can use it to model resources having complex setup operations. Even if the setup times are not significant enough to be included in the model, due to process complexity and safety, it is not desirable to frequently change the configuration of these resources. Most of the time, the complexity of the resuming setup time is related only to a subset of resources, while the others can be easily preempted and resumed without significant impact. We can then declare the resources with a significant setup as non-preemptive, and those with an insignificant setup as preemptive. In this way, we can better manage the preemption over activities with significant setup; at the same time, we delete the subjectivity of choosing the penalty (M_i) needed in our previous approach. Indeed, choosing the right penalty is

	Duration	Skill needed	Machine	Dead line	Release date	Activity type
Act 1	4	(S1,1)	(M1,1)	-	-	Preemptive
Act 2	2	(S3,1),(S4,1)	(M1,1)	5	3	Non-preemptive
Act 3	4	(S2,1)	(M1,1)	-	-	Partially preemp (M1 can not be released)

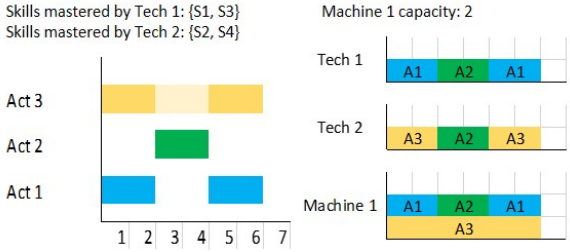

Figure 3. Example of an MSPSP-PP instance.

a tricky job. If M_i is too big, there is no interest in allowing the preemption of the activity, what may increase the $C_{\max}$ of the project. Conversely, if M_i is too small, this could cause activities to be preempted too many times disturbing the correct development of activities.

Including the concept of partial preemption in traditional scheduling problems may allow a reduction in the $C_{\max}$. In fact, for almost every scheduling problem in the scientific literature if an activity requires a non-preemptive resource, then this activity must be handled as uninterruptible , which produces an increase in the $C_{\max}$. Partial preemption acquires prominent importance when we schedule activities having a narrow time window or when resource availability varies over time. Knowing that preemptive versions give better values of $C_{\max}$, we can then establish the following relation for makespan value:

$$C_{\max}(Preemptive) \leq C_{\max}(Partially\ preemptive) \leq C_{\max}(Non\text{-}preemptive)$$

The complexity of the MSPSP with partial preemption can be established using the classical RCPSP as a starting point. For each instance of the RCPSP, we can match an instance of the MSPSP with partial preemption, where all resources are mono-skilled, and none of the resources can be preempted. Thus, we can define the RCPSP as a particular case of the MSPSP with partial preemption. Since the RCPSP has been proved to be strongly NP-hard (Błazewicz et al. 1983), we can, therefore, infer that the MSPSP with partial preemption is also strongly NP-hard.

We present below two formulations for the MSPSP-PP using Mixed-Integer/Linear Programming (MILP) and Constraint Programming (CP). Early versions of these formulations are presented in (Polo-Mejía et al. 2018).

4.1. *Mixed-Integer/Linear Programming (MILP)*

For modelling the MSPSP-PP, we use the standard 'On/Off' formulation again. Most of the constraints are similar to those proposed for the MSPSP with penalty for preemption. The main difference lies in the way we handle the preemption. Before, we wanted to know how many times an activity was preempted. For the MSPSP-PP we must identify all periods over which the activity remains stopped. To do this, in the MILP formulation presented by Polo-Mejía et al. (2018), one uses a set of auxiliary step binary variables indicating the end and start time of activities. This time, we

Table 3. Variables for the MSPSP-PP.

Notation for variables	
$Y_{i,t} \in \{0,1\}$	$Y_{i,t} = 1 \iff$ activity $i \in I$ is executed during time $t \in H$
$O_{j,i,t} \in \{0,1\}$	$O_{j,i,t} = 1 \iff$ technician $j \in J$ is allocated to activity $i \in I$ during time $t \in H$
$S_{j,i} \in \{0,1\}$	$S_{j,i} = 1 \iff$ technician $j \in J$ is assigned to execute activity $i \in N$
$Pp_{i,t} \in \{0,1\}$	$Pp_{i,t} = 1 \iff$ the activity $i \in PP$ is being preempted at time $t \in H$
$G_i \in \mathbb{R}_+$	Start time of activity $i \in I$
$F_i \in \mathbb{R}_+$	Finish time of activity $i \in I$
$C_{\max} \in \mathbb{R}_+$	Project makespan

reformulated the problem and decided to use continuous variables for determining the end and start time of each activity. This decision leads to a decrease in the number of variables and constraints needed by the model. The other difference with the MILP formulation of Polo-Mejía et al. (2018) is the presence of constraints (11,12) that ensure that a technician cannot be changed during the execution of a non-preemptive activity. Adding these constraints to the formulation proposed in Polo-Mejía et al. (2018) yields a formulation that is outperformed by the new formulation: on the instance set of 30 activities presented in Section 5, 80 instances out of 200 are solved to optimality by both formulation but the average time to reach optimality is significantly faster for the new formulation: 130s vs 184s on the same computer.

Notation for parameters. Additionally to the parameters described in Table 1, we also use an additional parameter $PR_{i,k} \in \{0,1\}$ that indicates whether the activity $i \in PP$ allows the release of resource $k \in K$ during the preemption periods or not. The parameter is equal to 0 if the resource can be released, 1 otherwise.

Variables. Variables used to model the Multi-Skill Project Scheduling Problem with partial preemption are shown in Table 3.

Objective Function. As mentioned in Section 3.1, our primary objective is to minimise the project makespan. The problem is then defined as follows:

$$\min(C_{\max}) \tag{14}$$

s.t. Constraints (3)-(7) and (11)-(13) remain unchanged. Constraints (2) and (8)-(10) are changed by:

$$\left(\sum_{i \in I} Y_{i,t} + \sum_{i \in PP} PR_{i,k} * Pp_{i,t}\right) * br_{i,k} \leq DR_{k,t} \quad \forall k \in K, \forall t \in H \tag{15}$$

$$Pp_{i,t} \leq 1 - Y_{i,t} \qquad \forall i \in PP, \forall t \in H \tag{16}$$

$$Pp_{i,t} \leq \sum_{t'=1}^{t} Y_{i,t'} \qquad \forall i \in PP, \forall t \in H \tag{17}$$

$$Pp_{i,t} \leq \sum_{t'=t}^{T} Y_{i,t'} \qquad \forall i \in PP, \forall t \in H \qquad (18)$$

$$F_i - G_i + 1 \leq D_i + \sum_{t \in H} Pp_{i,t} \qquad \forall i \in PP \qquad (19)$$

$$F_i - G_i + 1 \leq D_i \qquad \forall i \in N \qquad (20)$$

$$F_i \geq t * Y_{i,t} \qquad \forall i \in I, \forall t \in H \qquad (21)$$

$$G_i \leq t * Y_{i,t} + (1 - Y_{i,t}) * T \qquad \forall i \in I \qquad (22)$$

$$Y_{i,t},\ O_{j,i,t},\ S_{j,i},\ Pp_{i,t} \in \{0,1\}$$
$$C_{\max},\ G_i,\ F_i, \in \mathbb{R}_+$$

Constraints (15) ensure the respect of resource needs during the execution periods ($Y_{i,t} = 1$) and also the respect of needs for non-preemptive resources ($PR_{i,k} = 1$) during the preemption periods ($Pp_{i,t} = 1$). With constraints (16) we indicate that $Pp_{i,t}$ must be zero if the activity $i \in PP$ is in execution at time t ($Y_{i,t} = 1$). Constraints (17) and (18) ensure that the $Pp_{i,t}$ variables are always equal to zero outside its execution periods. Constraints (19) ensure that $Pp_{i,t}$ take value of 1 for periods where the activity i has been preempted. Constraints (20) guarantee that activities within N are not preempted. The finish time of each activity is calculated with Constraints (21). Constraints (22) calculate the start time of each activity.

4.2. *Constraint Programming (CP)*

Constraint programming is a technique coming from logic programming and artificial intelligence to solve complex combinatorial problems using a declarative description. The idea is to separate the constraint declaration using a rich constraint language from the solution-finding process based on the so-called constraint propagation, an active use of constraints to reduce the search space (Baptiste et al. 2001; Young et al. 2017). Constraint programming has been successfully applied to solve RCPSP. In (Kreter et al. 2017), the authors proposed and tested six different CP models for the RCPSP with calendar constraint, and they showed that CP solutions are highly competitive with existing MILP formulations of the problem (average runtime required by CP models is lower). Young et al. (2017) used pure Constraint Programming to solve the MSPSP. In their work, the authors proposed and tested different configurations of CP models, together with different search and propagation techniques. Using the best of their configurations, the authors were able to close an important number of open instances of the literature, showing a better performance than the Branch-and-Price algorithm proposed by Montoya et al. (2014), thus proving the interest of using CP for solving the MSPSP.

Given these promising results, we then propose to use CP for modelling the MSPSP with partial preemption. To model the MSPSP-PP, we use the software IBM CP Optimizer (CPO), making use of the concept of interval variables, a constrained object tailored to scheduling problems, and also to other specific scheduling constraints already defined in CPO (Laborie 2009).

In practice, there are two different ways to model preemptive activities using interval variables: either a chain of optional intervals of variable duration whose sum of durations is equal to the duration of activity (D_i); or a chain of D_i intervals of unitary

duration. In the CP formulation presented in Polo-Mejía et al. (2018), the authors used the first approach with the CPO `synchronize` function. We propose here to use the second way together with the CPO `alternative` function, which has better propagation than `synchronize`. These modifications lead to a significant improvement in the performance of the model[1]. On the set of 30-activity instances presented in Section 5, the previously proposed model was unable to solve any instance to optimality, while the new CP model solves to optimality 160 instances out of 200 in 85s in average.

Variables

Activities execution intervals: The $itvs_i$ interval variables will indicate the interval between the start and the end of each activity $i \in I$. For non-preemptive activities, the size of the interval must be equal to the duration of the activity (D_i). For preemptive and partially preemptive activities, the size of the intervals varies from D_i to T (the solver must decide the final size of the interval variable).

Intervals for each part of activities : In this model each preemptive ($i \in P$) and partially preemptive activity ($i \in PP$) is divided in D_i parts of unitary duration. The $par_{i,v}$ interval variable indicates the interval during which each unit of duration $v \in V_i$ ($V_i = \{1, .., D_i\}$) of activity $i \notin N$ is executed. For non preemptive activities we generate only one part ($V_i = \{1\}$) with size equal to the activity duration.

Technicians allocation: We made use of the optional interval variables proposed in CPO. Optional interval variables may or may not be present in the solution, so as to satisfy the constraints. The interval variable $InTech_{j,i,v}$ indicates the period when technician $j \in J$, if present, is working in the part $v \in V_i$ of activity $i \in I$.

Number of technicians allocated to each part: The integer variables $nTech_{i,v}$ indicate the number of technicians that are allocated to execute the part $v \in V_i$ of activity $i \in I$.

Number of technicians per skill allocated to each part: The integer variables $nSk_{s,i,v}$ indicate the number of technicians mastering the skill $s \in S$ allocated to the part $v \in V_i$ of the activity $i \in I$.

Objective function

The objective in the MSPSP-PP is to minimise the project makespan. This can be expressed in CPO as follows:

$$minimise\left(\max_{\forall i \in I}\{itvs_i.end\}\right)$$

Starting from the idea of trying to allocate always a minimum number of technicians to the activities, one can add a secondary criterion (lexicographic, using staticLex of

[1]We thank Dr Philippe Laborie, who helped us to improve the CP formulation.

CPO) that minimises the total number of technician allocations. The function staticLex defines a multi-criteria policy, ordering the different criteria and performing lexicographic optimisation. The first criterion is considered to be the most important, and any improvement of this criterion is worth any loss on the other criteria. The solver should be able to found better solutions faster when using the minimisation of the total number of technician allocations as a secondary objective. The objective function can then be defined as:

$$minimise\left(staticLex\left(\max_{\forall i \in I}\{itvs_i.end\}, \sum_{i \in I}\sum_{v \in \overline{V}} nTech_{i,v}\right)\right) \tag{23}$$

Constraints

Span(a, {b1, .., bn}) constraint states that the interval variable a (if present) spans over all present interval variables from the set b1, .., bn. In other words, interval variable a starts together with the first present interval from {b1, .., bn} and ends together with the last present interval. We use this kind of constraint to span the $par_{i,v}$ variables within the $itvs_i$ variables.

$$span(itvs_i, par_{i,v} : \forall v \in V_i)\ \forall i \in I \tag{24}$$

We also need to ensure that there is not overlapping for the parts of each activity. The predefined constraint `endBeforeStart(a,b)` indicates that the interval variable a must end before the interval variable b begins. These constraints are only necessary for preemptive and partially preemptive activities, and can be stated as follows:

$$endBeforeStart(par_{i,v}, par_{i,p})\ \forall i \notin N, \forall v \in V_i, \forall p \in V_i : p > v \tag{25}$$

Let us define $rUsage_k$ as a cumulative function indicating the usage of resource k over the time, and let DR_k be a cumulative function indicating the resource capacity over the time. Also let $pulse(A, h)$ be an elementary pulse function taking the value of h over the interval A. Preemptive resources ($PR_{i,k} = 0$) are used during the execution intervals of the parts ($par_{i,v}$). Non-preemptive resources ($PR_{i,k} = 1$), on the other hand, must be allocated during the whole execution interval of the activity ($itvs_i$). We can state the resource constraint as follows:

$$\begin{aligned} rUsage_k = & \sum_{i \in I : PR_{i,k}=0} \sum_{p} pulse(par_{p,i}, br_{i,k}) + \\ & \sum_{i \in I : PR_{i,k}=1} pulse(itvs_i, br_{i,k}) \quad \forall k \in K \end{aligned}$$

$$rUsage_k <= DR_k\ \forall k \tag{26}$$

We must guarantee that each technician is allocated at most to one activity at the time. For this, we use the predefined *noOverlap({b1, .., bn})* constraint that states that none of the interval variables within the set {b1, .., bn} overlap over the time. Note that we could use this expression for establishing the no overlap constraint of $Par_{i,v}$ variables (25). However, the way we declare it allows us to break some symmetries on the model. The disjunctive constraint over the technicians is then defined as:

$$noOverlap(InTech_{j,i,v} : \forall i \in I, \forall v \in V_i) \ \ \forall j \in J \tag{27}$$

Technicians cannot be assigned during their absence periods. To model these constraints, we define a step function describing the present and absent periods of each technician ($PreTech_j$). We must also use the predefined constraint *forbidExtent(a, Q)*. This expression states that whenever the interval variable a is present, it cannot overlap a point t where the step function Q(t) = 0. We ensure that absence/presence periods are respected as follows:

$$forbidExtent(InTech_{j,i,p}, PreTech_j) \, \forall j, \forall i, \forall p \tag{28}$$

For ensuring the skills requirements we use the expression $alternative(a, \{b_1, .., b_n\}, c)$. The alternative constraint will enforce that if a is present, then c and only c of the interval variable within $\{b_1, .., b_n\}$ will be present, and synchronised with a. In other words, c interval variables will be selected among the set and those c selected intervals will have to start and end together with interval variable a. We can use this constraint to select the technicians that will effectuate each skill for every part of an activity. It can be defined as:

$$\begin{aligned} alternative(par_{i,v}, InTech_{j,i,v} : \forall j \in J : CO_{j,s} = 1, nSk_{s,i,v}) \\ \forall i \in I, \forall v \in V_i, \forall s \in S \end{aligned} \tag{29}$$

The number of selected technicians for each skill and part goes from the skill requirement of the activity ($bs_{i,s}$) up to the maximum between the minimal number of required technicians (nt_i) and the sum of all the skill needs of the activity. Constraints (29) and (30) ensure the respect of skill requirements.

$$bs_{i,s} \leq nSk_{s,i,v} \leq \max \left\{ nt_i, \sum_{s' \in S} bs_{i,s'} \right\} \ \ \forall i \in I, \forall v \in V_i, \forall s \in S \tag{30}$$

We use the alternative constraint again to assure the respect of the minimal number of technicians:

$$alternative(par_{i,v}, InTech_{j,i,v} : \forall j \in J, nTech_{i,v}) \ \ \forall i \in I, \forall v \in V_i \tag{31}$$

The number of technicians allocated to each part of an activity will vary from the maximum between Nt and the highest skill requirement, up to the maximum

between nt_i and the sum of all skill requirements. Together with constraints (31), these constraints ensure the allocation of the minimal number of technicians.

$$max\left\{nt_i, \max_{\forall s \in S}\{bs_{i,s}\}\right\} \leq nTech_{i,v} \leq \max\left\{nt_i, \sum_{s \in S} bs_{i,s}\right\} \ \forall i \in I, \forall v \in V_i \tag{32}$$

The precedence relationships can be stated as:

$$endBeforeStart(itvs_i, itvs_l) \ \forall (i,l) \in E \tag{33}$$

Respect the deadlines and release dates are guarantee by:

$$itvs_i.end \leq dl_i \ \ \forall i \in I \tag{34}$$

$$r_i \leq itvs_i.start \ \ \forall i \in I \tag{35}$$

Since not all the technicians are available at the same time, we can add some redundant constraints to improve the lower bounds as well as the constraint propagation. Let the parameters $avTec_t$ be the amount of available technicians during period t, and $avSk_{c,t}$ the number of available technicians at time t mastering skill s. We define two cumulative functions $TecUsage$ and $SkUsage_c$ indicating the number of technicians allocated over the time, and the number of technicians mastering skill s allocated over the time respectively. We get the value for these functions as follows:

$$TecUsage = \sum_{i \in I} \sum_{v \in V_i} pulse\left(par_{i,v}, \max\left\{nt_i, \max_{\forall s \in S}\{bs_{i,s}\}\right\}\right)$$

$$SkUsage_s = \sum_{i \in I} \sum_{v \in V_i} pulse(bs_{i,s}) \ \ \forall s \in S$$

The $alwaysIn(F, B, min, max)$ constraint is used to confine the values of a cumulative function F during an interval $[u, v)$ inside interval [min,max]. We can then limit the number of technician allocated at each period t as follows:

$$alwaysIn(TecUsage, t, t+1, 0, avTec_t) \ \forall t \in H \tag{36}$$

$$alwaysIn(SkUsage_s, t, t+1, 0, avSk_{s,t}) \ \forall t \in H, \forall s \in S \tag{37}$$

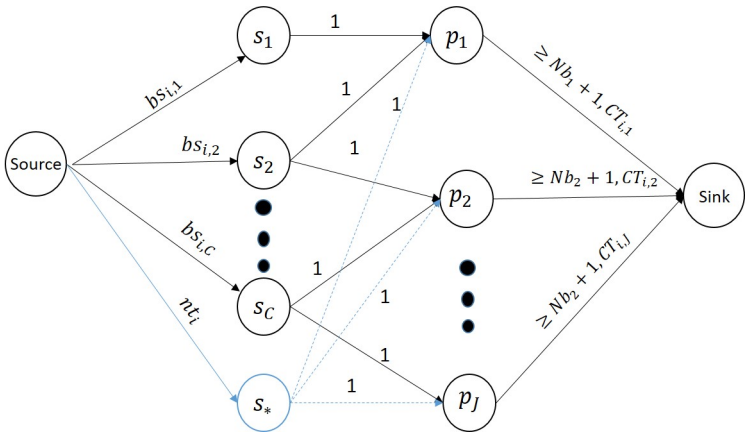

Figure 4. Flow graph for the allocation problem of the MSPSP-PP.

4.3. *Greedy algorithm for the MSPSP-PP*

As indicated in Section 2, we must be able to propose good solutions in short times to answer the industrial needs. The mathematical and logic models presented in Section 4 may not answer to this requirement for industrial-size instances (more than 100 activities for the LECA-STAR). In fact, when trying to solve such instances with the MILP model, the solver runs out of memory (15 GB of RAM). The CP model scales better, but the solver has some issues to prove optimality, so the question arises whether better feasible solutions could be obtained faster. In this section, we briefly describe a greedy algorithm for finding good quality solutions in reasonably fast computational times. A more detailed description of the heuristic can be found in Polo-Mejía et al. (2019).

The MSPSP-PP can be seen as a problem consisting of two coupled subproblems: an activity scheduling problem combined with an allocation problem of the technicians performing each activity. In a heuristic approach, once the order in which the activities will be executed is defined, we still have the problem of choosing the technicians who will perform them. To achieve this allocation in the best way, we must first allocate the technicians with the least chances of being necessary to the activities not yet scheduled, that is to say, the less critical technicians.

For this heuristic method, we propose to use a serial schedule generation scheme with priority rules. At each iteration of the greedy algorithm, one chooses one activity following a priority rule. We identify then the earliest period t where this activity can be scheduled (according to its preemption type, and resource and technicians availability). One can model the allocation problem as a Minimum-Cost Maximum-Flow (MCMF) problem (Ahuja 2017), as in the approach proposed by Bellenguez-Morineau (2006) in her PhD dissertation for the (non-preemptive) MSPSP, where the cost flow is related to technician criticality and the analysed activity. $G_{i,t} = (X, F)$, $X = S_i \cup P_t$ (Figure 4), where S_i represents the set of skills required by activity i and P_t is the subset of technicians available during period t (for preemptive and partially preemptive activities) or from t up to $t + D_i - 1$ (for non preemptive activities). F is the set of arcs connecting the nodes. The MCMF problem aims at minimising the cost required to deliver the maximum amount of flow possible in the network.

In this graph, we connect all vertices $s_c \in S_i$ to the source vertex with arcs having a maximum capacity equal to the requirement of skill c for executing activity i ($bs_{i,s}$).

To ensure the minimal number of technicians (nt_i), we add an additional vertex s_* linked to the source vertex with a capacity equal to nt_i and connected to all vertices $p_j \in P_t$ with a capacity of 1. Each $s_c \in S$ is linked to the all all vertices $p_j \in P_t$ of technicians mastering the skill c. These arcs have a maximum capacity of 1 to ensure that each technician responds to no more than one unit of need per skill. Finally, all vertices in P_t are connected to the sink vertex whit arcs having the capacities of at least the number of skills mastered by the technician j plus one ($Nb_j + 1$, to take into account the additional skill of "being a technician"). These are the only arcs having an associated cost ($CT_{i,j}$) related to the criticality of the technician p_j and the activity i that is being scheduled (see Definition 4.1).

Definition 4.1. The criticality cost $CT_{i,j}$ of a technician j is an indicator of the degree to which a technician could be requested by the set of not yet scheduled activities (set L). Let us define ST_j as the skill set that a technician j masters, and let SA_i be the skill set needed to execute the activity i. The criticality of a technician is directly proportional to the sum of duration (D_l) of every activity $l \in L$ multiplied by the $Cardinality(ST_j \cap SA_i)$. This indicator is calculated as follows:

$$CT_{i,j} = \frac{\sum_{\forall l \in L} (D_l * Cardinality(ST_j \cap SA_l))}{Cardinality(ST_j \cap SA_i)} \tag{38}$$

In case of an equality of such a cost for different technicians, we break the ties to ensure that the flow algorithm always minimises the number of technicians allocated to each activity.

Using one of the existing polynomial algorithms, the Edmonds-Karp algorithm (Edmonds and Karp 1972), for example, one can solve the problem of maximum flow at minimum cost for the proposed graph. To determine the technicians to allocate, we look at the vertices $p_j \in P_t$ through which the flow passes.

Algorithm 1: Greedy Serial Scheme Generation

(1) Select an activity from the list
(2) Find the earliest periods when this activity can be scheduled according to: its preemption type, and resources and technicians availability
(3) Allocate the technicians solving the MCMF problem:
- For non-preemptive activities the flow problem is solved only once (same technicians must execute the whole activity)
- For preemptive and partially preemptive activities, we must solve the flow problem for each unit of duration of the activity

(4) Return to Step 1 if there are still activities to be scheduled. Stop otherwise

The process for determining if an activity can be scheduled at time t varies according to its preemption type. For non-preemptive activities, we identify first the set of technicians that are available for all periods from t up to $t + D_i - 1$. If the sum of selected technicians mastering each skill is larger or equal than the skill need, and the number of selected technicians is larger or equal than Nt_i, and all the required cumulative resources are available, we say that the activity can start at period t, and we solve the MCMF problem. For partially preemptive activity, we will first

determine the minimum *end date* for the activity, starting from the analysed t period, depending on the availability of preemptive resources and technicians at each period. We will then check the continuous availability of non-preemptive resources from t to *end date*. If non-preemptive resources are available without interruption, we allocate them to the activity from t until *end date*. We solve the MCMF problem for periods $t \leq t' \leq$ *end date* where all preemptive resources are available. For non-preemptive activities, we allocate each unit of duration independently, so we solve the MCMF problem if all resources requirement can be satisfied during the period t.

The presence of deadlines is one of the critical constraints for generating feasible solutions using heuristic methods. In order to maximise the chance of finding feasible solutions, we propose to use a 2-step approach to generate the schedule. As a first step, activities with a deadline and its predecessor (to ensure the respect of precedence relationships) are scheduled following a *slack time-based* (Definition 4.2) priority list. Activities with a lower margin are scheduled first.

Definition 4.2. "Slack time" ($Slack_i$) refers to the margin that an activity i has in its planning window. It is a function of the deadline (dl_i), the earliest start time ($rmin_i$), and the activity duration (D_i). We calculate it as follows:

$$Slack_i = dl_i - rmin_i - D_i \tag{39}$$

Once planned activities with a deadline and its predecessors, we must perform the scheduling of the remaining activities (L). To choose the order in which activities will be scheduled, we propose to use the most common priority rules in the scheduling literature:

- Longest Duration (LD): prioritises the activity i with the greatest duration (D_i).
- Most Successors (MS): prioritises the activity i with the highest number of successors.
- Earliest Start Time (EST): prioritises the activity i with the lowest earliest start date ($rmin_i$).
- Earliest Finish Time (EFT): prioritises the activity i with the smallest "earliest finish time". This date is calculated by adding the duration of the activity (D_i) to the earliest start date ($rmin_i$), ie: $rmin_i + D_i$.
- Greatest Rank (GR): prioritises the activity A_i for which the sum of the duration of its successors is the largest.
- Greatest Resource Demand (GRD): prioritises the activity i with the highest resource consumption.

An additional priority rule can be generated by solving the continuous relaxation of the MILP model proposed in Section 4.1. Activities with the lowest start date in the relaxed problem will be scheduled first. We can also generate priority rules using the continuous relaxation of the MILP for the MSPSP with penalty for preemption (MSPSP-wP) presented in Section 4.1. For this, we transform each MSPSP-PP instance into a MSPSP-wP instance, where the partially preemptive activities become preemptive activities with penalties. This transformation can be seen as a Lagrangian relaxation of the partial preemption constraint (we use different values of M_i during the computational test).

In order to increase the chances of finding a feasible solution from the beginning, and even improve the solution, we propose to build the set of activities to schedule L as follows: $L = \{N, PP, P\}$ where N is the subset of non-preemptive activities, PP is the subset of partially preemptive activities and P is the subset of preemptive activities. N, PP and P are sorted according to the priority rule. With this approach, we exploit the ability of preemptive and partially preemptive activities to fill the unused spaces left after scheduling the non-preemptive activities.

The heuristic presented before is a single-pass heuristic because only one priority rule is used to select the activities to be scheduled. In order to improve the results we get, we can execute the procedure using all the priority rules presented before and keeping the minimum makespan. This process originates a so-called *multi-pass* heuristic.

5. Experimental results for the MSPSP-PP

For computational tests, we use a computer equipped with an Intel Xeon E5-2695 processor at 2.3 GHz. We use CPLEX 12.7 and CP Optimizer 12.7 for solving the MILP models and the CP model respectively (using the default configuration and limiting the number of threads used by the solvers at 8). We generated sets of instances using a generation algorithm to obtain realistic data w.r.t. the case-study and that allows fixing aspects such as proportions of preemption type, percentage of activities with time windows, density of precedence relationships, skill number per technician, etc.

5.1. *Performance of the MILP and CP models*

To test the behaviour of our models regarding the proportion of each activity type (preemptive, partially preemptive and non-preemptive) present in an instance, we generated four sets (A1, B1, C1 and D1) of 50 instances. Table 4 presents the specific distribution of the preemption type for each set. All instances have 30 activities with a duration between 5 to 10 time units, up to 15 skills (following a uniform distribution between 3 and 15) , up to 8 cumulative resources, 20% of activities with time windows. 8 technicians (multi-skilled resources) are available in 2 teams (4 technicians by team) doing work-shift of 12 hours each. The number of skill master for each technician follows a uniform distribution between 5 and 10 skill. Only 10% of the activities in each instances has precedence constraints and the maximal number of predecessor is 3. The remainder characteristics are randomly generated. Since time-indexed formulations require an initial estimation of the scheduling horizon, we initially tested the two models using the sum of activities duration as the scheduling horizon.

Table 4. Distribution of preemption types per set of instances.

	Set A1	Set B1	Set C1	Set D1
Non-preemptive	10%	10%	80%	33.3%
Partially preemptive	10%	80%	10%	33.3%
Preemptive	80%	10%	10%	33.3%

Table 5 presents, for a computation time limited to 10 minutes, the number of instances for which each model found at least one feasible solution, the number of instances whom optimality was proven, the average time required to prove the opti-

mality, and the average gap, percentage difference between the $C_{\max}$ value of the best found solution and the optimal $C_{\max}$ (the optimal solution was obtained by running the MILP solver over a computing platform until proving the optimality). One can see that the CP model outperforms the MILP formulation, being able to found feasible solutions for all the instances, and proving the optimality of a larger number of instances. If we analyse the individual results of each model regarding the preemption type, one sees that the presence of a high percentage of non-preemptive activities decreases, even more, the performance of the MILP model (the model only get 4 out of 50 feasible solutions for the Set C1). This can be explained by the fact that more auxiliary variables and constraints are needed for this kind of activities. The CP model, on the other hand, seems to perform better when this portion is high. This is because a lower number of interval variables ($par_{i,v}$ and $InTech_{j,i,v}$) are needed, thus a lower number of decisions are to be made.

Table 5. Results of MILP and CP models after 10 min of computation

	MILP model				CP model			
	Instances with initial solution	Instances solved to optimality	Average time to optimality	Average gap	Instances with initial solution	Instances solved to optimality	Average time to optimality	Average gap
Set A1	48	16	335.2 s	40.63%	50	37	180.32 s	0.16%
Set B1	25	0	-	30.97%	50	33	141.92 s	0.43%
Set C1	4	0	-	195%	50	37	107.58 s	0.72%
Set D1	29	0	-	97.9%	50	33	161.43 s	0.44%
All	106	16	335.2 s	57.92%	200	140	147.59	0.44%

One can initialise the solver with an initial solution (warm start) obtained by a heuristic method. We test both formulations using an initial solution generated using the greedy algorithm presented in Section 4.3. The use of warm start improves the performance of both models significantly (see Table 6). For the CP model, the average gap and the average time required to prove the optimality are reduced in almost a half, compared with the results without warm start. For instances within the set A1, the average gap was not reduced, but the time required to prove the optimality of the instances was reduced. For instances in set C1, on the other hand, the average time to optimality did not change, while the average gap was reduced to a half. For sets B1 and D1, both the average time to optimality and the average gap were reduced. Analysing results from Table 6, one can see that MILP outperforms CP when the percentage of preemptive activities is high (set A1). CP, on the other hand, gives better results when this percentage is low. One could then say that the two methods are complementary. Future research should be done in order to develop a hybrid method that better exploit the characteristics of each instance.

Table 6. Results of MILP and CP models after 10 min of computation using warm start

	MILP			CP		
	Number of instances solved to optimality	Average time to optimality	Average gap	Number of instances solved to optimality	Average time to optimality	Average gap
Set A1	46	87.39 s	0.05%	39	67.17 s	0.18%
Set B1	15	154.12 s	2.69%	40	88.01 s	0.15%
Set C1	0	-	9.45%	41	108.73 s	0.39%
Set D1	19	216.12 s	1.99%	40	76.14 s	0.21%
All	80	130.48 s	3.55%	160	85;27 s	0.23%

5.2. *Greedy algorithm*

For testing the performance of the heuristic method, we generated new sets of instances (A2, B2, C2, D2). For each instance on a set, there is an instance on the other sets having the same characteristics, except for the distribution of preemption type for

the activities (see Table 4). Each of the four sets has a total of 50 instances, each of them with 50 activities to be scheduled, and an expected $C_{\max}$ going from 130 to 170 time units. The average duration of the activities goes from 5 up to 15 time units. All other characteristics are similar to the ones presented before for set 1. We decide to use instances with only 50 activities, instead of 100 activities that are in average the number of activities scheduled at the LECA-STAR every week, to have better lower bounds for evaluating the optimality gap of the proposed heuristics. Indeed the solvers (MILP and CP) may have some issues for finding good lower bounds for larger instances. The proposed heuristics has been coded in C++. To solve the flow problems, we used the adapted C++ version of the Edmonds-Karp algorithm proposed by Ababei (2009). To obtain the lower bound or, in some cases, the optimal solutions, we use the MILP and CP models proposed in Section 4.

Table 7 shows the average gap values (percentage difference between the obtained solution an the best known lower bound obtained by the CP or MILP solver after 2 hours of computing) for the greedy algorithm using the priority rules presented in Section 4.3. It also shows the average gap for the CP model after 5 minutes of computation using CP Optimizer with only one thread. Results obtained with the MILP model are not given since the solver run out of memory before giving any initial solution. For the priority rule based on the continuous relaxation of the MILP for the MSPSP-wP, in a matter of simplification, we decided that the value of penalty will be the same for all activities ($M_i = M$). We tested the heuristics using different values of M: 0, 0.5, 1, and 1.5. We observe that the heuristic using Greatest Resource Demand (GRD) as priority rule seems to give the lower average gap, followed by the priority rules Longest Duration (LD) and Most Successors (MS). The worst results are obtained using the Earliest Start Time (EST) and Earliest Finish Time (EFT).

Table 7. Average gap for the greedy algorithm per priority rule

	Gap for the greedy algorithm				
	Set A2	**Set B2**	**Set C2**	**Set D2**	**All**
LD	7.33%	7.78%	15.65%	9.28%	10.01%
MS	8.44%	8.26%	16.85%	9.27%	10.70%
EST	9.61%	9.99%	18.98%	10.00%	12.14%
EFT	10.68%	10.79%	22.72%	10.48%	13.67%
GR	8.69%	8.49%	16.66%	9.28%	10.78%
GRD	7.33%	7.88%	15.90%	8.02%	9.78%
Relax MSPSP-PP	8.90%	8.95%	17.27%	9.70%	11.20%
Relax MSPSP-wP $M_i = 0$	8.90%	9.88%	17.95%	9.77%	11.63%
Relax MSPSP-wP $M_i = 0.5$	9.23%	9.15%	19.40%	10.02%	11.95%
Relax MSPSP-wP $M_i = 1$	9.05%	9.14%	19.08%	9.79%	11.77%
Relax MSPSP-wP $M_i = 1.5$	9.39%	9.23%	19.45%	9.87%	11.99%
Multi-pass	5.30%	5.90%	12.26%	6.21%	7.42%
CP (after 5 min)	6.01%	6.65%	7.65%	5.56%	6.47%

If we compare the results of the multi-pass version of the greedy algorithm against the results obtained after 5 minutes of computing of the CP model, we see that the average gap obtained by CP is slightly lower than the one obtained by the greedy algorithm. However, if we analyse the results for each set of instances, we observe that the greedy algorithm gets a lower average gap for instances with a low proportion of non-preemptive activities (Sets A2 and B2). Statistical tests indicate that the CP model (after 5 min) statistically outperforms the greedy algorithm only when the proportion of non-preemptive activity is high (Set C2). Note that the computation time for obtaining the greedy solution is lower than one second (15 seconds for the priority rule derived from the linear relaxation), which proves the interest of the greedy algorithm.

6. Conclusions and future research

The primary objective of this paper was to propose a way to apply operations research techniques to schedule research activity within a nuclear facility. Since we are working over a short scheduling horizon, we can decompose each research activities into a series of elementary tasks, and use traditional models to schedule them. The use of scheduling models represent an improvement of the facility safety and also allows researchers to save time, that before was used to planning activities, and devote it to research.

We showed the need to modify classical scheduling models to adapt them to a real-life problem. We also showed how these adapted or extended versions might need to be improved even more to represent the same industrial problem better. In our specific case, an initial analysis of the characteristics of the research facility, led us to propose a Multi-Skill Project Scheduling Problem with penalty for preemption (for which we have proposed a MILP formulation) in the first instance. However, a more in-depth analysis showed us that this approach only fulfilled the operational and technical requirements of the research facility partially. We then propose a more accurate model: the MSPSP with partial preemption. The concept of partial preemption leads to a limited release of the resources during the preemption periods. We modelled this problem using two different techniques: Mixed-Integer/Linear Programming and Constraint Programming.

The MILP formulation showed an outstanding performance in the presence of a high proportion of preemptive or partially preemptive activities. However, it starts having troubles to find initial solutions and prove the optimality when the proportion of non-preemptive activities increases. The CP formulation, on the other hand, presented the opposite behaviour; it performs better when the instances are highly non-preemptive. This behaviour leads us to think that the two modelling techniques could be complementary. As future research, we must then study better ways to combine and exploit the advantages of both techniques.

The MILP and CP models presented in this paper could be not fast enough to generate good quality solutions in short time for large instances. Having good solutions in reduced time is essential for the industrial application of this problem. That is why we also present a greedy algorithm aiming to answer the industrial need. This heuristic allowed us to obtain average optimality gap of 7.42% within a few seconds.

As future work, we should focus our efforts on identifying ways to improve the models proposed in this article such as: reformulation to get a better quality of the linear relaxation for the MILP models and breaking symmetries in the space search for the CP model. A theoretical study of the polyhedral characteristics of the MILP models should also be done. Due to the industrial application aspect of our research project, we must also develop new problem dependant (meta)heuristics or matheuristics to ensure we can get good solutions faster.

References

Ababei, C. (2009, December). C++ adapted version of the Edmonds-Karp relabelling MCMF algorithm. https://github.com/eigenpi/MCMF4. Online; accessed 01 September 2018.

Ahuja, R. K. (2017). *Network Flows: Theory, Algorithms, and Applications* (1st ed.). Pearson Education.

Almeida, B. F., I. Correia, and F. Saldanha-da Gama (2019). Modeling frameworks for the multi-skill resource-constrained project scheduling problem: a theoretical and empirical com-

parison. *International Transactions in Operational Research 26*(3), 946–967.

Artigues, C. (2008). The resource-constrained project scheduling problem. In *Resource-constrained project scheduling: models, algorithms, extensions and applications*, pp. 21–36. John Wiley & Sons.

Baptiste, P., C. Le Pape, and W. Nuijten (2001). *Constraint-based scheduling*. Boston/Dordrecht/London: Kluwer Academic Publishers.

Bellenguez-Morineau, O. (2006). *Méthodes de résolution pour un problème de gestion de projet multi-compétence*. Ph. D. thesis, Université Franois Rabelais, Tours.

Bellenguez-Morineau, O. (2008, March). Methods to solve multi-skill project scheduling problem. *4OR 6*(1), 85–88.

Błazewicz, J., J. K. Lenstra, and A. H. Rinnooy Kan (1983). Scheduling subject to resource constraints: classification and complexity. *Discrete applied mathematics 5*(1), 11–24.

Certa, A., M. Enea, G. Galante, and C. Manuela La Fata (2009). Multi-objective human resources allocation in r&d projects planning. *International Journal of Production Research 47*(13), 3503–3523.

Chen, R., C. Liang, D. Gu, and J. Y. Leung (2017). A multi-objective model for multi-project scheduling and multi-skilled staff assignment for it product development considering competency evolution. *International Journal of Production Research 55*(21), 6207–6234.

Chen, S.-K., Y.-W. Ti, and K.-Y. Tsai (2016). Nuclear power plant construction scheduling problem with time restrictions: a particle swarm optimization approach. *Science and Technology of Nuclear Installations 2016*.

Correia, I., L. L. Lourenço, and F. Saldanha-da Gama (2012). Project scheduling with flexible resources: formulation and inequalities. *OR spectrum 34*(3), 635–663.

Correia, I. and F. Saldanha-da Gama (2015). A note on branch-and-price approach for the multi-skill project scheduling problem. *Optimization Letters 9*(6), 1255–1258.

Dhib, C., A. Soukhal, and E. Néron (2015). Mixed-integer linear programming formulation and priority-rule methods for a preemptive project staffing and scheduling problem. In *Handbook on Project Management and Scheduling Vol. 1*, pp. 603–617. Springer.

Dupin, N. and E.-G. Talbi (2016). Dual heuristics and new lower bounds for the challenge euro/roadef 2010. *Matheuristics 2016*, 60.

Edmonds, J. and R. M. Karp (1972, April). Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM) 19*(2), 248–264.

Gavareshki, M. K. (2004). New fuzzy gert method for research projects scheduling. In *2004 IEEE International Engineering Management Conference (IEEE Cat. No. 04CH37574)*, Volume 2, pp. 820–824. IEEE.

Hassanzadeh, F., M. Modarres, H. R. Nemati, and K. Amoako-Gyampah (2014). A robust R&D project portfolio optimization model for pharmaceutical contract research organizations. *International Journal of Production Economics 158*, 18–27.

Jost, V. and D. Savourey (2013). A 0–1 integer linear programming approach to schedule outages of nuclear power plants. *Journal of Scheduling 16*(6), 551–566.

Kazemipoor, H., R. Tavakkoli-Moghaddam, and P. Shahnazari-Shahrezaei (2013). Solving a novel multi-skilled project scheduling model by scatter search. *South African Journal of Industrial Engineering 24*(1), 121–131.

Khemakhem, M. A. and H. Chtourou (2013). Efficient robustness measures for the resource-constrained project scheduling problem. *International Journal of Industrial and Systems Engineering 14*(2), 245–267.

Klein, R. (2000). Project scheduling with time-varying resource constraints. *International Journal of Production Research 38*(16), 3937–3952.

Kreter, S., A. Schutt, and P. J. Stuckey (2017). Using constraint programming for solving RCPSP/max-cal. *Constraints 22*(3), 432–462.

Laborie, P. (2009, May). IBM ILOG CP Optimizer for detailed scheduling illustrated on three problems. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, Lecture Notes in Computer Science, pp. 148–162. Springer, Berlin, Heidelberg.

Li, H. and K. Womer (2009). Scheduling projects with multi-skilled personnel by a hybrid MILP/CP benders decomposition algorithm. *Journal of Scheduling 12*(3), 281.

Maghsoudlou, H., B. Afshar-Nadjafi, and S. T. A. Niaki (2018). Preemptive multi-skilled resource constrained project scheduling problem with hard/soft interval due dates. *RAIRO-Operations Research*.

Mancel, C. (2004). *modélisation et résolution de problèmes d'optimisation combinatoire issus d'applications spatiales*. Ph. D. thesis, INSA de Toulouse.

Montoya, C., O. Bellenguez-Morineau, E. Pinson, and D. Rivreau (2014). Branch-and-price approach for the multi-skill project scheduling problem. *Optimization Letters 8*(5), 1721–1734.

Néron, E. (2002). Lower bounds for the multi-skill project scheduling problem. In *Proceeding of the Eighth International Workshop on Project Management and Scheduling*, pp. 274–277.

Norouzi, G., M. Heydari, S. Noori, and M. Bagherpour (2015, June). Developing a mathematical model for scheduling and determining success probability of research projects considering complex-fuzzy networks. *Journal of Applied Mathematics 2015*(Article ID 809216). 15 pages.

Pérez, F., T. Gómez, R. Caballero, and V. Liern (2018). Project portfolio selection and planning with fuzzy constraints. *Technological Forecasting and Social Change 131*, 117–129.

Petersen, G. M. (2016). *Algorithms and Methods for Optimizing the Spent Nuclear Fuel Allocation Strategy*. Ph. D. thesis, University of Tennessee, Knoxville, USA.

Polo-Mejía, O., M.-C. Anselmet, C. Artigues, and P. Lopez (2017, October). A new RCPSP variant for scheduling research activities in a nuclear laboratory. In *47th International Conference on Computers & Industrial Engineering (CIE47)*, Lisbon, Portugal, pp. 463–470.

Polo-Mejía, O., M.-C. Anselmet, C. Artigues, and P. Lopez (2018, June). Mixed-integer and constraint programming formulations for a multi-skill project scheduling problem with partial preemption. In *12th International Conference on Modelling, Optimization and Simulation (MOSIM 2018)*, Toulouse, France, pp. 367–374.

Polo-Mejía, O., C. Artigues, and P. Lopez (2019, February). A heuristic method for the multi-skill project scheduling problem with partial preemption. In *8th International Conference on Operations Research and Enterprise Systems*, Prague, Czech Republic, pp. 111–120.

Reyck, B. D. and R. Leus (2008). R&D project scheduling when activities may fail. *IIE Transactions 40*(4), 367–384.

Young, K. D., T. Feydy, and A. Schutt (2017, August). Constraint programming applied to the Multi-Skill Project Scheduling Problem. In *Principles and Practice of Constraint Programming*, Lecture Notes in Computer Science, pp. 308–317. Springer, Cham.