ISSN: 0020–7543

INTERNATIONAL JOURNAL OF

Production Research

Official Journal of the International Foundation for Production Research

Taylor & Francis
Taylor & Francis Group

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

Journal:	*International Journal of Production Research*
Manuscript ID	TPRS-2019-IJPR-0551
Manuscript Type:	Special Issue Paper
Date Submitted by the Author:	17-Mar-2019
Complete List of Authors:	Polo-Mejia, Oliver; CEA, DEC, SETC; LAAS, ROC Artigues, Christian; Université de Toulouse, LAAS-CNRS Lopez, Pierre; LAAS-CNRS, ROC
Keywords:	SCHEDULING, LINEAR PROGRAMMING, CONSTRAINT PROGRAMMING
Keywords (user):	Multi-skill research project, MSPSP, Nuclear facility, Partial preemption

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

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

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

ARTICLE HISTORY
Compiled March 17, 2019

ABSTRACT
This paper presents the first results of a research project aiming to optimise scheduling of activities within one of the research laboratories of the 'Commissariat à l'Énergie Atomique et aux Énergies Alternatives (CEA)'. To tackle this problem, we decompose every research activity into a set of elementary tasks to apply standard scheduling methods. We can then model this scheduling problem as a more general version of the Multi-Skill Project Scheduling Problem. As a first approach, we propose a Multi-Skill Project Scheduling Problem with penalty for preemption, along with its mixed-integer/linear programming formulation, where the preemption is allowed applying a penalty every time an activity is interrupted. However, further analysis leads us to propose a more accurate variant of the problem: the Multi-Skill Project Scheduling Problem with partial preemption. This version is modelled using two methodologies: mixed-integer/linear programming and constraint programming. We also show how the variant with penalty for preemption can be used as a heuristic for the variant with partial preemption.

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

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

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

ARTICLE HISTORY
Compiled March 17, 2019

ABSTRACT
This paper presents the first results of a research project aiming to optimise scheduling of activities within one of the research laboratories of the 'Commissariat à l'Énergie Atomique et aux Énergies Alternatives (CEA)'. To tackle this problem, we decompose every research activity into a set of elementary tasks to apply standard scheduling methods. We can then model this scheduling problem as a more general version of the Multi-Skill Project Scheduling Problem. As a first approach, we propose a Multi-Skill Project Scheduling Problem with penalty for preemption, along with its mixed-integer/linear programming formulation, where the preemption is allowed applying a penalty every time an activity is interrupted. However, further analysis leads us to propose a more accurate variant of the problem: the Multi-Skill Project Scheduling Problem with partial preemption. This version is modelled using two methodologies: mixed-integer/linear programming and constraint programming. We also show how the variant with penalty for preemption can be used as a heuristic for the variant with partial preemption.

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

1. Introduction

The applications of operations research techniques for optimising experimentation activities are mainly related to deterministic project management techniques like the Critical Path Method (CPM) (Bodea et al. 2011). However, more complex techniques have been developed to have a better representation of the research project uncertainty. These techniques can use a stochastic project scheduling approach like in the Program Evaluation and Review Technique (PERT) or the Graphical Evaluation and Review Technique (GERT); or a fuzzy project scheduling approach, when the statistical information is hard to find, like in the Fuzzy GERT (FGERT) (Karimi Gavareshki 2004) and, the more recent, Parallel and Reversible-Fuzzy GERT (PR-FGERT) (Norouzi et al. 2015).

All the techniques mentioned above focus mainly on planning the project macro-tasks. However, in our study, we are interested in scheduling research activities at the operational level. More precisely, we work on the modelling and optimisation of the weekly scheduling within one of the nuclear research facilities of the 'Commissariat à l'Énergie Atomique et aux Énergies Alternatives (CEA)', a French public government-funded research organisation. At operational level (short scheduling horizon), activities can be easily characterised thus reducing uncertainty. We use the elementary tasks ap-

proach proposed in Polo-Mejía et al. (2017), where we decompose every experiment or research activity into a series of elementary tasks that have well-defined characteristics. After this decomposition, we can use classical scheduling methods to model and optimise the weekly schedule within the research facility.

This paper is an extension of the works presented in Polo-Mejía et al. (2017) and Polo-Mejía et al. (2018), increasing the sample size for experimental results and also showing how the Multi-Skill Project Scheduling Problem with penalty for preemption can be used as a way to get faster and better solutions of the Multi-Skill Project Scheduling Problem with partial preemption.

The remainder of this paper is structured as follows: In the next section, we present some related problems that are the bases for modelling our case study. In Section 2 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 mixed-integer/linear programming formulation for the Multi-Skill Project Scheduling Problem with penalty for preemption. In Section 5, we present a more accurate variant we call Multi-Skill Project Scheduling Problem with partial preemption, followed by its mixed-integer/linear and constraint programming formulations. A method to solve the Multi-Skill Project Scheduling Problem with partial preemption using the Multi-Skill Project Scheduling Problem with penalty for preemption is also presented in this section. Experimental results are presented in Section 6. Finally, we conclude in Section 7.

2. Industrial problem description

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

The MSPSP is a particular case of the 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. We must find a feasible schedule minimising the project makespan while satisfying the precedence relationships and the resource constraints: a resource can execute only those skills it master and must answer to only one skill per activity (Bellenguez-Morineau 2008). The characteristics of the basic version of the problem are presented in Figure 1. We must effectuate some changes over the baseline version of MSPSP to get a most accurate representation of the industrial scheduling problem.

The most significant change is related to the possibility of interrupting an activity once this has started. The non-preemption constraint (activities execution must be without interruption) is one of the main characteristics of the MSPSP (Néron 2002). For technical reasons, we can not guarantee the continuous execution of activities with a duration larger than the work shift since some resources may be absent during the night (activity splitting, see Cheng et al. (2015)), and also some times we need to give priority to critical activities. Allowing the preemption is then necessary for modelling our problem. However, there is a subset of particular activities (set $\overline{P}$) for which safety and operational constraints force us to execute them without interruption.

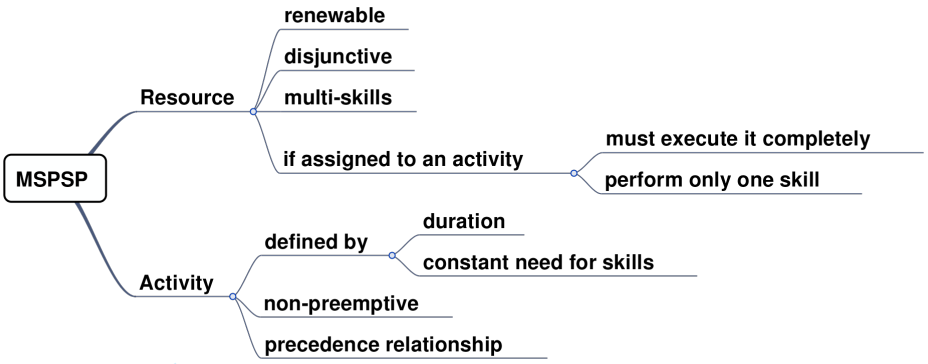


Figure 1. Characteristics of the Multi-Skill Project Scheduling Problem (MSPSP).

Most of the preemptive scheduling problems assume that activities can be interrupted and resumed later without any consequence (Ballestín et al. 2008). This assumption does not represent industrial reality (Ballestín et al. 2009; Vanhoucke 2008), because of the presence of setup costs or times needed for resuming the activity or also because of the reduction in the production rate. To address this issue, Afshar-Nadjafi (2014) proposed an RCPSP with penalty for preemption, where the author applies a penalty every time an activity is interrupted. In our facility, nuclear regulation requires that, for 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. At first, a suitable approach would be to use an MSPSP with penalty for preemption where we apply a penalty (M_i) every time an activity (A_i), for which we want to limit the preemption, is preempted.

However, experimental tests with real data and exchanges with researchers of the laboratory showed that this approach does not fulfil all industrial needs (see Section 4). A more accurate modelling consists in introducing the concept of 'partial preemption', allowing the release of only a part of the resources during the preemption periods. This yields the MSPSP with partial preemption (MSPSP-PP). To the best of our knowledge, this is the first time the concept of partial preemption is proposed.

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. capacity bigger than 1 activity at the time) mono-skilled resources (compound machines and buildings). Our technicians can execute several skills at the time per activity. However, we must ensure a minimal number of technicians to comply with safety and operational constraints. Additionally, since some activities have duration larger than technicians work shifts, the technicians can partially execute the activities, except for non-preemptive activities, $i \in \overline{P}$, 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). Representing the absences of technicians over time is crucial in our problem; thus the MSPSP-PP works with time-varying resource capacity.

Traditional MSPSP does not take into account time windows for scheduling the

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

Notation for parameters	
$DR_{k,t}$	Capacity of resource k at time t
$Br_{i,k}$	Requirement of resource k for executing activity i
$DO_{j,t}$	Availability of technician j at time t. Equal to 1 if available, 0 otherwise
$CO_{j,c}$	Indicates whether a technician j masters skill c or not. Equal to 1 if mastered, 0 otherwise
$Bc_{i,c}$	Needs of skill c for executing activity i
Nt_i	Minimal number of technicians required to execute activity i
D_i	Duration of activity i
dl_i	Deadline for activity i
r_i	Release date for activity i
M_i	Penalty for preempting activity i
I	Set of activities to be scheduled
J	Set of available technicians
C	Set of skills
K	Set of mono-skilled resources
$\overline{P}$	Set of non-preemptive activities
E	Set of precedence relationships

activities. We must, however, include time windows to our problem. Nuclear regulation, for example, requires to carry out a series of periodic tests to ensure the proper functioning of the machines. These test must be scheduled and executed after a fixed r_i, but before a deadline (dl_i). Once defined all the characteristics of the industrial problem, we can now begin the description of its mathematical formulation.

3. MILP for the MSPSP with penalty for preemption

The RCPSP and the MSPSP variants can be modelled by different families of integer (ILP) or mixed-integer (MILP) linear programming formulations: continuous time-based models based on flows (Artigues et al. 2003), discrete-time ILP formulations (Artigues 2017), valid antichain based model (Damay et al. 2007), or event-based MILP formulations (Koné et al. 2011). After a deep analysis of all these options, we conclude that a discrete-time formulation is the most accurate approach to model the problem at hand (especially for modelling the time-varying resources availability and activity preemption). This formulation known as 'on/off' uses binary variables $Y_{i,t}$, where $Y_{i,t} = 1$ if activity i is in progress at time t and $Y_{i,t} = 0$ otherwise (Artigues 2017). This formulation has been identified as the most accurate for modelling the time-varying resources 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.

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 is in progress at time t
$O_{j,i,t} \in \{0,1\}$	$O_{j,i,t} = 1 \iff$ technician j is allocated to activity i at time t
$Al_{j,i} \in \{0,1\}$	$Al_{j,i} = 1 \iff$ technician j executes at least one unit of duration of activity i
$X_{i,t} \in \{0,1\}$	$X_{i,t} = 1 \iff$ activity i is started or resumed at time t
$C_{\max} \in \mathbb{Z}_+$	Project makespan

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, we must be sure that the set of weekly activities is 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:

$$min(C_{\max} + \sum_{\forall i \in I} \sum_{t=1}^{T} M_i * X_{i,t}) \tag{1}$$

s.t.

$$\sum_{i \in I} O_{j,i,t} \leq DO_{j,t} \qquad \forall j \in J, \forall t \in 1..T \tag{2}$$

$$\sum_{i \in I} (Y_{i,t} * Br_{i,k}) \leq DR_{k,t} \qquad \forall t \in 1..T, \forall k \in K \tag{3}$$

$$Y_{i,t} * Bc_{i,c} \leq \sum_{j \in J} (O_{j,i,t} * CO_{j,c}) \forall i \in I, \forall t \in 1..T, \forall c \in C \tag{4}$$

$$\sum_{j \in J} O_{j,i,t} \geq Y_{i,t} * Nt_i \qquad \forall t \in 1..T, \forall i \in I \tag{5}$$

$$\sum_{t=1}^{T} Y_{i,t} \geq D_i \qquad \forall i \in I \tag{6}$$

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

$$\sum_{t=dl_i+1}^{T} Y_{i,t} \leq 0 \qquad \forall i \in I \tag{8}$$

$$\sum_{t=1}^{r_i-1} Y_{i,t} \leq 0 \qquad \forall i \in I \tag{9}$$

$$X_{i,t} \geq Y_{i,t} - Y_{i,t-1} \qquad \forall i \in I, \forall t \in 2..T \tag{10}$$

$$X_{i,1} = Y_{i,1} \qquad \forall i \in I \tag{11}$$

$$\sum_{t=1}^{T} X_{i,t} \leq 1 \qquad \forall i \in \overline{P} \tag{12}$$

$$O_{j,i,t} \geq Al_{j,i} + Y_{i,t} - 1 \qquad \forall j \in J, \forall t \in 1..T, \forall i \in \overline{P} \tag{13}$$

$$O_{j,i,t} \leq Al_{j,i} \qquad \forall j \in J, \forall t \in 1..T, \forall i \in \overline{P} \tag{14}$$

$$C_{\max} \geq t * Y_{i,t} \qquad \forall i \in I, \forall t \in 1..T \tag{15}$$

Constraints (2) ensure that technicians are allocated only if they are available.Resources and skills requirements are ensured in constraints (3) and constraints (4). The minimum number of technicians (Nt_i) and duration of activities (D_i) are given in constraints (5) and (6). Precedence relationships are ensured by (7). Constraints (8) and (9) limit the activity executions between their release dates (r_i) and deadlines (dl_i). Constraints (10) and (11) determine when an activity is preempted. In inequalities (12) we assure that preemption is not allowed for non-preemptive activities ($i \in \overline{P}$). Constraints (13) and (14) ensure that if a technician is allocated to a non-preemptive activity it must execute it until completeness. Finally, the makespan of the project is calculated using constraints (15).

4. MSPSP with partial preemption (MSPSP-PP)

As mentioned in the introduction, after testing the MSPSP with penalty for preemption using real data and sharing the results with the employees of the research facility, we realised that the model only fulfilled the operational requirements partially. Indeed, choosing the right penalty (M_i) is 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. However, if M_i is too small, this could cause activities to be preempted too many times disturbing the correct development of activities.

The possibility of releasing all resources during preemption periods does not represent the real behaviour for some activities within the nuclear research facility. Let us take the example of activities requiring an inert atmosphere for their executions. These activities require that the equipment ensuring the inert atmosphere must be allocated until the activity is complete without any interruption, while the other resources can be released without any problem if the activity is interrupted. Until now, the only way to model this, while respecting safety requirements, was to declare this kind of activities as "non-preemptive". However, this decision can increase the project's makespan. Aiming to overcome this inconvenience, we propose a new variant of the MSPSP that better represents the behaviour of our laboratory: the MSPSP with partial preemption (MSPSP-PP).

In the MSPSP-PP, if an activity is preempted, we release only a subset of resources while seizing the remainders (*partial preemption*). We can then can then classify the activities in three types according to the possibility of releasing the resources during the preemption periods: 1) Non-preemptive activities, if none of the resources can be released; 2) Partially preemptive activities, if a subset of resources can be released;

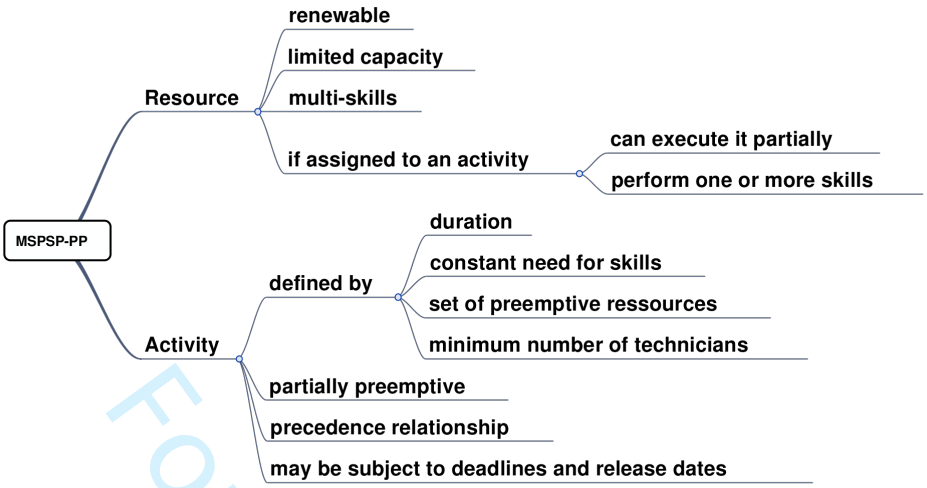


Figure 2. Characteristics of the Multi-Skill Project Scheduling Problem with Partial Preemption (MSPSP-PP).

and 3) Preemptive activities, if all resources can be set free. All other characteristics are the same as those presented in Section 2 and are summarised in Figure 2.

Using this concept, 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. 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 (MSPSP with penalty for preemption).

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 non-preemptive, which produces an increase in the $C_{\max}$. Experimental results (not presented in this document) show us that 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:

$$C_{\max}(Preemp.) \leq C_{\max}(Partially\,preemp.) \leq C_{\max}(Non\text{-}preemp.)$$

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

Table 3. Additional parameters for the MSPSP-PP.

Notation for parameters	
$PR_{i,k}$	Indicates whether the activity i allows the release of resource k during the preemption periods or not. Equal to 0 if the resource can be released, 1 otherwise
Pc_i	Indicates whether the activity i allows the release of technicians during the preemption periods or not. Equal to 0 if the technicians can be released, 1 otherwise

Table 4. Variables for the MSPSP-PP.

Notation for variables	
$Y_{i,t} \in \{0,1\}$	$Y_{i,t} = 1 \iff$ activity i is in progress at time t
$O_{j,i,t} \in \{0,1\}$	$O_{j,i,t} = 1 \iff$ technician j is allocated to activity i at time t
$Al_{j,i} \in \{0,1\}$	$Al_{j,i} = 1 \iff$ technician j executes at least one unit of duration of activity i
$Z_{i,t} \in \{0,1\}$	$Z_{i,t} = 1 \iff$ activity i starts at time t or before
$W_{i,t} \in \{0,1\}$	$W_{i,t} = 1 \iff$ activity i ends at time t or after
$Pp_{i,t} \in \{0,1\}$	$Pp_{i,t} = 1 \iff$ activity i is preempted during period t
$C_{\max} \in \mathbb{Z}_+$	Project makespan

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 (Section 3). However, to handle the partial preemption, we must change the way we model the preemption periods.

Notation for parameters. Parameters used for this model are identical to those presented in Table 1. However, we must add the parameters shown in Table 3.

Variables. Variables used to model the Multi-Skill Project Scheduling Problem with partial preemption are shown in Table 4.

Objective Function. As mentioned in Section 3, our primary objective is to minimise the project makespan. The problem is then defined as follows:

$$\min(C_{\max}) \tag{16}$$

s.t.

$$\sum_{i \in I} O_{j,i,t} \leq DO_{j,t} \qquad \forall j \in J, \forall t \in 1..T \tag{17}$$

$$\sum_{i \in I} ((Y_{i,t} + PR_{i,k} * Pp_{i,t}) * Br_{i,k}) \leq DR_{i,k} \qquad \forall t \in 1..T, \forall k \in K \tag{18}$$

$$(Y_{i,t} + Pc_i * Pp_{i,t}) * Bc_{i,c} \leq \sum_{j \in J} (O_{j,i,t} * CO_{j,c}) \quad \forall i \in I, \forall t \in 1..T, \forall c \in C \tag{19}$$

$$\sum_{j \in J} O_{j,i,t} \geq (Y_{i,t} + Pc_i * Pp_{i,t}) * Nt_i \qquad \forall t \in 1..T, \forall i \in I \qquad (20)$$

$$\sum_{t=1}^{T} Y_{i,t} \geq D_i \qquad \forall i \in I \qquad (21)$$

$$D_l * (1 - Y_{i,t}) \geq \sum_{t'=t}^{T} Y_{l,t'} \qquad \forall (i,l) \in E, \forall t \in 1..T \qquad (22)$$

$$\sum_{t=dl_i+1}^{T} Y_{i,t} \leq 0 \qquad \forall i \in I \qquad (23)$$

$$\sum_{t=1}^{r_i-1} Y_{i,t} \leq 0 \qquad \forall i \in I \qquad (24)$$

$$Z_{i,t} \geq Y_{i,t'} \qquad \forall i \in I, \forall t \in 1..T, \forall t' \in 1..t \qquad (25)$$

$$W_{i,t} \geq Y_{i,t'} \qquad \forall i \in I, \forall t \in 1..T, \forall t' \in t..T \qquad (26)$$

$$Pp_{i,t} = Z_{i,t} + W_{i,t} - Y_{i,t} - 1 \qquad \forall i \in I, \forall t \in 1..T \qquad (27)$$

$$O_{j,i,t} \geq Al_{j,i} + Y_{i,t} + Pp_{i,t} - 1 \qquad \forall j \in J, \forall t \in 1..T, \forall i \in \overline{P} \qquad (28)$$

$$O_{j,i,t} \leq Al_{j,i} \qquad \forall j \in J, \forall t \in 1..T, \forall i \in \overline{P} \qquad (29)$$

$$C_{\max} \geq t * Y_{i,t} \qquad \forall i \in I, \forall t \in 1..T \qquad (30)$$

All constraints from (17) to (24) have identical functions than those presented in Section 3: Constraints (17) ensure the respect of technicians availability; with constraints (18), (19) and (20) we ensure that the needs of resources, skills and minimal number of technicians are satisfied during the execution periods ($A_{i,t} = 1$) and during the preemption periods ($Pp_{i,t} = 1$) for non-preemptive resources and technicians; constraints (21) enforce each activity to be in process according to its duration; precedence relationships constraints are given in (22); constraints (23) and (24) are related to deadlines and release dates. Constraints (25) force the auxiliary binary variable $Z_{i,t}$ to be equal to 1 for all periods equal or greater than the start date of the activity. Constraints (26), on the other hand, force the auxiliary binary variable $W_{i,t}$ to be equal to 1 for all periods equal or lower than the completion time of the activity. Using constraints (27) we determine the periods during which an activity has been preempted (or not). Constraints (28) and (29) state that all technicians allocated to a non-preemptive activity must execute it until its completeness. Finally, constraints (30) express the project's total duration.

Note that in this model we do not explicitly state a constraint limiting the preemption for non-preemptive activities, as this is implicitly done by the model (Constraints (18), (19) and (20)) and the objective function.

4.2. *Constraint Programming (CP)*

Constraint programming is an emergent technique to solve complex combinatorial problems using a declarative description, and it comes from logic programming and artificial intelligence. The idea is to separate the constraint declaration using a rich constraint language from the solution finding process based on an active use of constraints to reduce the search space (constraint propagation) (Baptiste et al. 2001; Young et al. 2017). This paradigm has been used successfully to solve various schedul-

Table 5. Interval variables for the MSPSP-PP.

Notation for variables	
$itvs_i$	Interval variable between the start and the end of activity i
$par_{i,p}$	Optional interval variable indicating the start and the end of every possible part p, $p \in \{1, 2, \ldots, D_i\}$, of activity i
$InTech_{j,i,p}$	Optional interval variable indicating the periods when technician j is working in the part p of activity i

ing problems (Baptiste et al. 2001; Gökgür et al. 2018), obtaining very good 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 define in CPO (Laborie 2009). A more detailed version of this model can be found in Polo-Mejía et al. (2018).

Variables The used variables are presented in Table 5. Note that an optional interval variable may or may not be present in the solution, which has to be decided during the solving process. Namely, the $InTech_{j,i,p}$ variable is not present if technician j is not assigned to part p of activity i. Similarly, the D_i intervals variables $part_{i,p}$ represent the maximum preemption of an activity into D_i parts. If the activity is less preempted, some interval variables $part_{i,p}$ shall be absent.

Objective function The minimisation of the $C_{\max}$ is equivalent to minimise the end date ($itvs_i.end$) of the last activity. We must formulate our objective function as follows:

$$\min(\max_{\forall i \in I} itvs_i.end) \tag{31}$$

Constraints The first thing we must ensure is that, for every activity i, all interval variables $par_{i,p}$ (representing the activity parts) are within the span of the interval variable $itvs_i$ (representing the whole activity execution interval independently of the preempted portions). For this, we make use of the predefined *Span(a,* $\{b_1, .., b_n\}$*)* constraint. This function states that the interval variable a starts at the same time that the first present interval variable from $\{b_1, .., b_n\}$ and ends together with the last present interval (Laborie 2009). The constraint is declared as:

$$span(itvs_i, par_{i,p} : \forall p) \ \forall i \in I \tag{32}$$

If a technician j must execute the activity i during the interval $par_{i,p}$, then the start and end of interval variable $InTech_{j,i,p}$ must be synchronised with $par_{i,p}$. This constraint can be stated with the *synchronize(a,* $\{b_1, ..., b_n\}$*)* function , that forces the start and end of interval variables $\{b_1, ..., b_n\}$ to be the same than in interval variable a (if a is present):

$$synchronize(par_{i,p}, InTech_{j,i,p} : \forall j) \ \forall i \in I, \forall p \in P \tag{33}$$

For describing the disjunctive capacity of technicians, we must insure that for each technician j its interval variables $InTech_{j,i,p}$ do not overlap. We use the *noOverlap(*$\{b_1, .., b_n\}$*)* that forbids interval variables within the set $\{b_1, .., b_n\}$ to over-

lap over time:

$$noOverlap(InTech_{j,i,p} : \forall i, \forall p) \ \ \forall j \in J \tag{34}$$

Interval variables $par_{i,p}$, of each activity A_i, cannot overlap either. However, to break symmetries, we do not use the $noOverlap$ expression, and we state the constraints as follows:

$$par_{i,p}.end < par_{i,s}.start \ \forall i \in I, \forall p \in P, \forall s \in Parts \ : \ s > p \tag{35}$$

$$presenceOf(par_{i,s}) \leq presenceOf(par_{i,p}) \ \forall i \in I, \forall p \in P, \forall s \in Parts \ : \ s > p \tag{36}$$

$presenceOf()$ is a Boolean constraint, taking a true value if the interval variable is present, false otherwise. To reduce even the search space, we use again this constraint to ensure that interval variables $InTech_{j,i,p}$ exists only if $par_{i,p}$ is present:

$$presenceOf(InTech_{j,i,p}) \leq presenceOf(par_{i,p}) \forall i \in I, \forall p \in P, \forall j \in J \tag{37}$$

To model technicians availability over time we use a step function $PreTech_j$, this function is equal to 1 for the interval where the technician is available, and it is equal to 0 for the remainders. For ensuring that technicians are not allocated during their absent periods, we use the $forbidExtent(a, F)$ constraint, that prevents interval variable a, if present, to overlap a point t where the step function $F(t) = 0$:

$$forbidExtent(InTech_{j,i,p}, PreTech_j) \, \forall j \in J, \forall i \in I, \forall p \in P \tag{38}$$

The $sizeOf$ expression returns the size (duration) of an interval variable. Te duration constraint of activities can be declared as follows:

$$D_i = \sum_{\forall p \in P} sizeOf(par_{i,p}) \ \forall i \in I \tag{39}$$

Scheduling time windows can be stated as:

$$dl_i \geq itvs_i.end \, \forall i \in I \tag{40}$$

$$r_i \leq itvs_i.start \, \forall i \in I \tag{41}$$

Precedence relationships can be modelled using the $endBeforeStart(A, B)$ expression. It ensures that interval variable A ends before the start of interval variable B. The constraint is then:

$$endBeforeStart(itvs_i, itvs_l) \ \forall (i, l) \in E \tag{42}$$

For declaring the capacity constraint for cumulative resources, we use a cumulative function $rUsage_k$ representing the usage of resource k over time. The $pulse(F, h)$ is

an elementary pulse function that is equal to h over interval F, and 0 for all other intervals:

$$rUsage_k = \sum_{i \in I: PR_{i,k}=0} \sum_p pulse(par_{p,i}, Br_{i,k}) + \sum_{i \in I: PR_{i,k}=1} pulse(itvs_i, Br_{i,k}) \ \forall k \in K$$

$$rUsage_k \leq DR_k \ \forall k \in K \tag{43}$$

We define $SkAll_{i,c}$ as a cumulative function determining the allocation of skill c for the activity i over time.

$$SkAll_{i,c} = \sum_j \sum_p pulse(InTech_{j,i,p}, CO_{j,c}) \ \forall i \in I, \forall c \in C$$

For ensuring that the allocated skills are always at least equal to the skills needs ($Bc_{i,c}$), we call the $alwaysIn(F, B, min, max)$ constraint. This constraint ensures that the values of a cumulative function (F) are always within $[min, max]$ during the interval B. For activities with preemptive technicians, we will declare this constraint over the intervals of real execution of the activity ($par_{i,p}$). However, if technicians are declared as not preemptive, the constraint is applied to the interval variable of the whole activity ($itvs_i$). The constraints are defined as follows:

$$alwaysIn(SkAll_{i,c}, par_{i,p}, Bc_{i,c}, \infty) \ \forall i : Pc_i = 0, \forall c \in C, \forall p \in P \tag{44}$$

$$alwaysIn(SkAll_{i,c}, itvs_i, Bc_{i,c}, \infty) \ \forall i : Pc_i = 1, \forall c \in C \tag{45}$$

We use a similar approach for ensuring the minimal number of technicians. We define first a cumulative function $AllTech_i$ that indicate the number of technicians allocated to an activity over time, and then we use the $alwaysIn$ constraint:

$$AllTech_i = \sum_j \sum_p pulse(InTech_{j,i,p}, 1) \ \forall i \in I$$

$$alwaysIn(AllTech_i, par_{i,p}, Nt_i, \infty) \ \forall i : Pc_i = 0, \forall p \in P \tag{46}$$

$$alwaysIn(AllTech_i, itvs_i, Nt_i, \infty) \ \forall i : Pc_i = 1 \tag{47}$$

Finally, we must ensure that if a technician is allocated to a non-preemptive activity ($i \in \overline{P}$), he must execute it during its entire duration. In order to do this, we must ensure that the activity is only divided in one part using the constraint *synchronize(a, $\{b_1, ..., b_n\}$)*.

$$synchronize(itvs_i, par_{i,1}) \ \forall i \in \overline{P} \tag{48}$$

4.3. *Heuristic based on MILP with penalty and flow*

It is clear that the MSPSP-PP gives us a better representation of the real-life scheduling process in the nuclear research facility. However, the time required to solve optimally industrial instances can be prohibitive. In our case, we must schedule every week about 50 activities with duration varying from a few hours to a couple of days. The set of skills an activity may need is formed by at least 20 skills, and we must allocate at least 15 technicians per week. MILP and CP formulations have shown to be inefficient to solve the research facility instances optimally in reduce time. This encourages us to propose a heuristic method for finding good solutions quicker.

Experimental results show us that, for instances having similar characteristics, the MSPSP with penalty for preemption can be solved faster than the MSPSP-PP. We can exploit this characteristic and use the MSPSP with penalty for preemption as a way to get good answers faster. We can transform every instance of the MSPSP-PP into an instance of the MSPSP with penalty for preemption where partially preemptive activities have a penalty $M_i > 0$ ($M_i = 0$ for all others). We can see this transformation as a Lagrangian relaxation of the MSPSP-PP where the 'non-release' constraint for partially preemptive activities is relaxed and included in the objective function. Note that this approach cannot be seen as a lower bound for the MSPSP-PP since, for avoiding the penalty, the model may sometimes give answers with a $C_{\max}$ higher than the optimum for the MSPSP-PP. However, if needed, we can get a lower bound just setting all penalties to zero.

This transformation should allow getting, in a shorter time, good solutions (near to the optimum) that in most cases comply with all MSPSP-PP constraints. However, in some case, a quick repair is needed. This repair can be done using a greedy algorithm that extracts the activity sequence from the obtained solution. This sequence is used later to generate a valid schedule using a serial generation scheme, such as the standard one for the RCPSP (Kolisch and Hartmann 1999).

For the MSPSP and its variants, additionally to the sequence of the activities, we must also decide the set of technicians that execute each activity. The best technicians allocation is ensured by always allocating first the less critical technicians (the one being less necessary to the remainder activities). We can solve this allocation problem, for each activity, modelling it as a Minimum-Cost Maximum-Flow (MCMF) problem (Ahuja 2017), as in the approach proposed by Bellenguez-Morineau (2008) for the MSPSP, where the cost flow is related to technician criticality. We construct a bipartite graph $G_i = (X, F)$, $X = S \cup P$, having in the first level the set of skills (S) needed to execute the activity A_i and in the second level, the subset of available technicians (P) mastering at least one of the skills within S (Figure 3).

In this graph, we connect all vertices $S_k \in S$ to the source vertex with arcs having a maximum capacity equal to the requirement of skill k for executing activity A_i ($b_{i,k}$). To ensure the minimal number of technicians (Nt_i), we add a special S_k vertex (S_*) representing the skill of 'being a technician'; the capacity of this arc is equal to Nt_i. Each $S_k \in S$ is linked to the all P_j of technicians mastering the skill k. 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 are connected to the sink vertex whit arcs having a maximum capacity equal to the number of skills the technician masters plus one (to take into account the additional skill of "being a technician") (Nb_j). These are the only arcs having an associated cost ($CT_{i,j}$) related to the criticality of the technician P_j. To determine which technicians allocate, we solve the MCMF problem and identify the vertices $P_j \in P$ through which the flow passes.

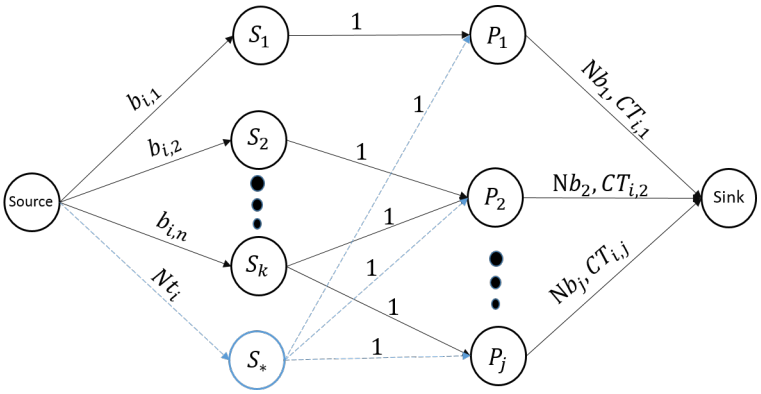


Figure 3. Flow graph for the allocation problem of the MSPSP-PP.

Table 6. Distribution of preemption types per set of instances.

	Set A	Set B	Set C	Set D
Non-preemptive	10%	10%	80%	33.3%
Partially preemptive	10%	80%	10%	33.3%
Preemptive	80%	10%	10%	33.3%

Time windows are always a critical constraint for finding feasible solution using a serial generation scheme. To overcome this problem, we can schedule at the beginning the activities having time windows, following a slack time-based priority list (activities with the lower slack time first). After, we schedule the remainder activities using the sequence extracted from the relaxed ILP solution. We must indicate that, even when using this approach, it may happen that we do not get a feasible solution, which was not the case for the instances used for the computational tests presented in Section 5.

5. Experimental results

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 4 sets (A1, B1, C1 and D1) of 42 instances. Table 6 presents the specific distribution of the preemption type for each set. All instances have 10 activities with a duration between 1 to 10 time units, 15 skills, 8 cumulative resources, 8 technicians (multi-skilled resources), 20% of activities with time windows. The remainder characteristics are randomly generated.

As indicated in Polo-Mejía et al. (2018), the use of a time-index formulation causes

Table 7. Solved instances per preemption type (time limited to 10 min).

		Set A1	Set B1	Set C1	Set D1	All
MILP	Solved to opt.	39	41	42	41	163
	Not solved to opt.	3	1	0	1	5
CP	Solved to opt.	25	32	42	32	131
	Not solved to opt.	17	10	0	10	37

Table 8. Time required to solve to optimality small instances.

	Average time to optimality		t-test
	MILP	CP	p-val
Set A1	1.70 sec	23.18 sec	0.045
Set B1	5.46 sec	11.5 sec	0.127
Set C1	44.26 sec	15.06 sec	0.0086
Set D1	8.47 sec	24.58 sec	0.217

the performance of our MILP model to be affected by the initial estimate of the total duration of the project (T). For these tests, we set T equal to the sum of all activities durations.

From Tables 7 and 8, we observe that the MILP model seems to outperform the CP model, what confirm the results presented in Polo-Mejía et al. (2018). The MILP model was able to solve to optimality more instances (163 out of 168) within the limited time (10 minutes) that the CP model which only was able to demonstrate the optimality for 131 instances. Results in Table 7 (showing the number of instances solved to optimality or not for each model) may suggest that the performance of the CP model is affected by the distribution of preemption type within the instances, being able to solve to optimality a more significant number of instances having a high portion of non-preemptive activities (Set C1).

We now analyse the time required to solve to optimality for those instances solved by both CP and MILP models. Table 8 shows the average time required to solve to optimality the instances according to their preemption type distribution; we also present the p-values of the t-test for paired samples used to determine whether the average times were statistically equal or not (for more details on this statistical test see Derrick et al. (2017)). These results confirm what we expected: the CP model is faster than the MILP model when the instance has a high percentage of non-preemptive activities (Set C1), but its performance decreases at the same that this percentage decrease. On the other hand, the MILP model has better performance for instances having a high percentage of preemptive activities (Set A1). For highly partially preemptive instances (Set B1) both models performances are statistically equivalent.

5.2. *Heuristic based on MILP with penalty*

To test the performance of the heuristic approach, we solve instances within set C1 using different values of preemption penalty (M) for partially preemptive activities (going from 0 to 1.5 with a step of 0.5). Table 9 summarises the obtained results. For all 42 instances within Set C1, we were able to get optimal solutions regardless of the penalty value. The time required to get these solutions is smaller than the one required when using MILP for MSPSP-PP (t-tests confirm that the solving times are statistically different). On the other hand, we do not get enough statistical evidence to affirm that the penalty value may or not have an impact over solving time.

For studying the behaviour of this approach over bigger instances, we generated 4 new sets (A2, B2, C2, and D2) of 50 instances each. Instances have a bigger number of

Table 9. Average optimality gap and time for for small instances (Set C1).

	Penalty value			
	0	**0.5**	**1**	**1.5**
Average time (s)	11.35	13.09	13.40	10.94
Average gap	0%	0%	0%	0%
p-val for time compared to MILP for MSPSP-PP	0.029	0.034	0.041	0.032

Table 10. Average optimality gap for bigger instances after 10 minutes.

		MSPSP with penalty of:			
	MSPSP-PP	0	0.5	1	1.5
Set A2	18.17%	0.69%	10.66%	11.32%	14.52%
Set B2	18.24%	0.77%	11.95%	13.03%	13.46%
Set D2	21.33%	1.13%	15.28%	14.00%	17.42%
All	18.84%	0.85%	12.13%	12.56%	14.68%

activities (50 activities) and a duration going from 5 to 10 time units (all other characteristics remaining unchanged from Section 5.1). As we want to use this approach for industrial application, we tested only the quality of the obtained answers after 10 minutes of computation (reasonable time to obtain a weekly schedule). Using the MILP model of the MSPSP-PP, we were able to obtain feasible solutions for only 116 instances out of 200 (no initial solution was found for the remaining instances within the limit time). We got feasible solutions for almost all instances within sets A2 and B2 (46 and 47 instances respectively). However, when the proportion of non-preemptive activities increases, the number of instances for which we get feasible solutions decreases (only 23 instances for the set D2 and none for the set C2). This confirms our previous conclusion: The MILP model performance is lower for instances with a high proportion of non-preemptive activities.

For testing the heuristic approach, we try to solve all instances of the second group using different values of preemption penalty (M) (going from 0 to 1.5 with a step of 0.5) and analysing the average gap (percentage difference between the heuristic solution and the optimal solution) we get for each of them. Table 10 summarises the results for the average gap. We observe that the heuristic, regardless of the penalty value, give us better results than the MILP model of the MSPSP-PP. We also observe that the heuristic method converges much faster to solutions close to the optimum when we do not use a penalty ($M = 0$). When we carry out the t-test for the average gap, we get that the heuristic using a penalty equal to 1.5 gives the worst results. This can be explained by the fact that with a higher penalty the interest of preempting activities decreases, which means an increase over the final $C_{\max}$. In addition to allowing a reduction in the average gap, the heuristic method allowed us to obtain initial solutions for 37 of the 84 instances for which the MILP of the MSPSP-PP did not find a solution within the time limit.

6. Conclusions and future research

The primary objective of this paper was to propose a way to apply operations research techniques to schedule research activities 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. This additionally to the decrease of the total duration of the projects.

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 deeper 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. We modelled this problem using two different techniques: Mixed-Integer/Linear Programming and Constraint Programming.

Computational test over the two modelling techniques for the MSPSP with partial preemption showed that the MILP model seems to outperform the CP model, being able to solve a more significant number of instances to optimality in the limited time. We also observed that the CP model is faster when the proportion of non-preemptive activities within an instances is high. The MILP model, on the other hand, gives better results when instances are highly preemptive.

Even if the MSPSP-PP gives us a better representation of the real-life scheduling process in the nuclear research facility, the time required to solve real-life instances to optimality can be prohibitive. That is why, we proposed a heuristic approach, using the MSPSP with penalty for preemption, to obtain better and faster results for the MSPSP-PP, compared to the use of the MILP model proposed in Section 4.1. In that direction, we could further improve our results and solve the real industrial application.

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. Because of 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

Afshar-Nadjafi, B. (2014, April). Resource constrained project scheduling subject to due dates: Preemption permitted with penalty. *Advances in Operations Research 2014*, 16–25.

Ahuja, R. K. (2017). *Network flows: Theory, algorithms, and applications* (1st ed.). Pearson Education.

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.

Artigues, C. (2017, March). On the strength of time-indexed formulations for the resource-constrained project scheduling problem. *Operations Research Letters 45*(2), 154–159.

Artigues, C., P. Michelon, and S. Reusser (2003, September). Insertion techniques for static and dynamic resource-constrained project scheduling. *European Journal of Operational Research 149*(2), 249–267.

Ballestín, F., V. Valls, and S. Quintanilla (2008, September). Pre-emption in resource-constrained project scheduling. *European Journal of Operational Research 189*(3), 1136–1152.

Ballestín, F., V. Valls, and S. Quintanilla (2009, November). Scheduling projects with limited number of preemptions. *Computers & Operations Research 36*(11), 2913–2925.

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

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

Bodea, C. N., I. R. Badea, and A. Purnus (2011, January). Distributed Research Project Scheduling Based on Multi-Agent Methods. *Journal of Applied Computer Science & Mathematics 5*(10), 20–26.

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.

Cheng, J., J. Fowler, K. Kempf, and S. Mason (2015). Multi-mode resource-constrained project scheduling problems with non-preemptive activity splitting. *Computers & Operations Research 53*, 275–287.

Damay, J., A. Quilliot, and E. Sanlaville (2007, November). Linear programming based algorithms for preemptive and non-preemptive RCPSP. *European Journal of Operational Research 182*(3), 1012–1022.

Derrick, B., D. Toher, and P. White (2017). How to compare the means of two samples that include paired observations and independent observations: A companion to Derrick, Russ, Toher and White (2017). *Tutorials in Quantitative Methods for Psychology 13*(2), 120–126.

Gökgür, B., B. Hnich, and S. Özpeynirci (2018). Parallel machine scheduling with tool loading: a constraint programming approach. *International Journal of Production Research 56*(16), 5541–5557.

Karimi Gavareshki, M. H. (2004, October). New fuzzy GERT method for research projects scheduling. In *2004 IEEE International Engineering Management Conference*, Volume 2, Singapore, pp. 820–824.

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

Kolisch, R. and S. Hartmann (1999). Heuristic algorithms for the resource-constrained project scheduling problem: Classification and computational analysis. In *Project scheduling*, pp. 147–178. Springer.

Koné, O., C. Artigues, P. Lopez, and M. Mongeau (2011, January). Event-based MILP models for resource-constrained project scheduling problems. *Computers & Operations Research 38*(1), 3–13.

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.

Néron, E. (2002). Lower bounds for the multi-skill project scheduling problem. In *Eighth International Workshop on Project Management and Scheduling*, Valencia, Spain.

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.

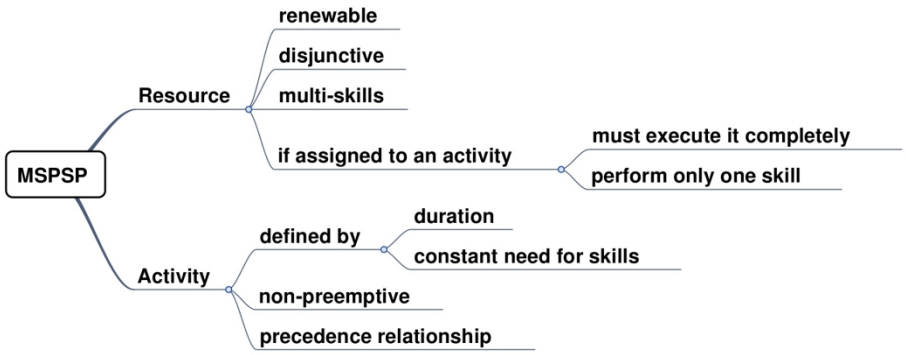
Polo-Mejía, O., M.-C. Anselmet, C. Artigues, and P. Lopez (2017, October). A new RCPSP variant for scheduling research activities in a nuclear laboratory. In *47th International Conference on Computers & Industrial Engineering (CIE47)*, 8 pages, Lisbon, Portugal.

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)*, 8 pages, Toulouse, France.

Vanhoucke, M. (2008, May). Setup times and fast tracking in resource-constrained project

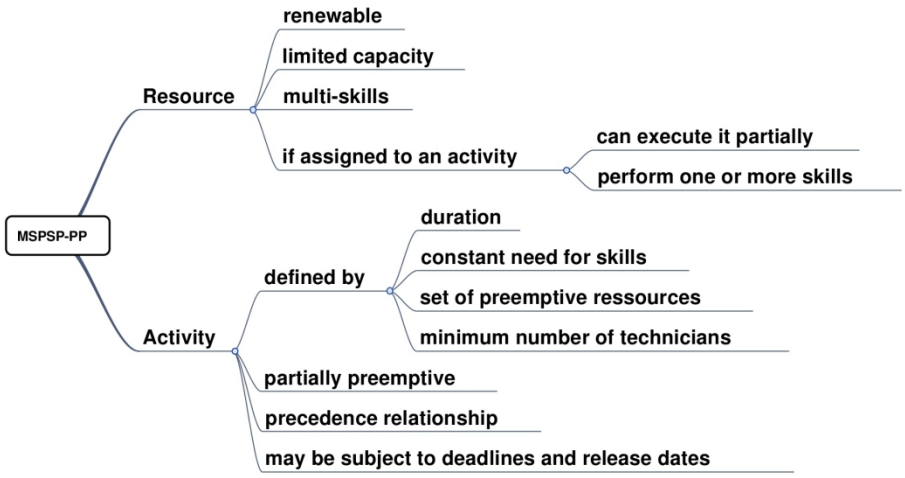
scheduling. *Computers & Industrial Engineering 54*(4), 1062–1070.

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.



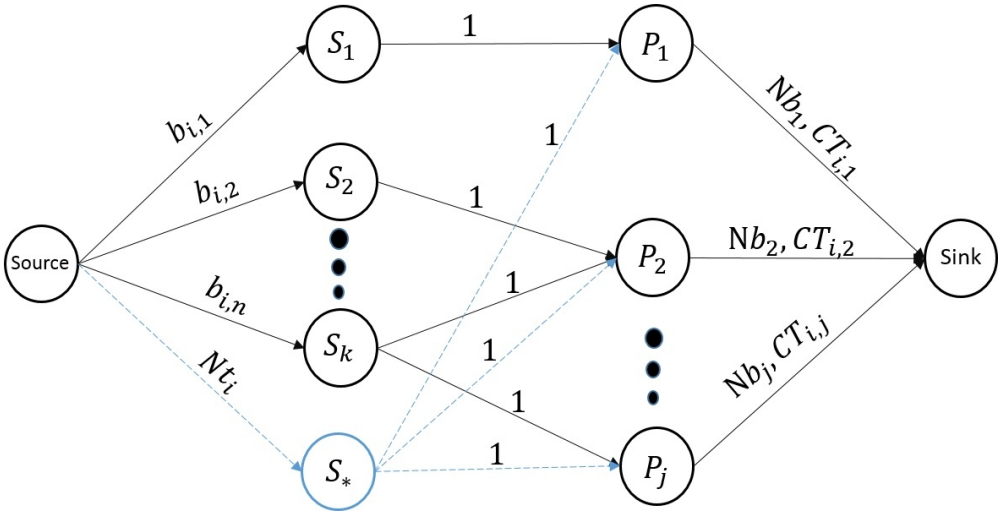
Characteristics of the Multi-Skill Project Scheduling Problem (MSPSP).

541x247mm (96 x 96 DPI)



Characteristics of the Multi-Skill Project Scheduling Problem with Partial Preemption (MSPSPPP).

541x313mm (96 x 96 DPI)



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

313x158mm (96 x 96 DPI)