Mixed-integer and constraint programming formulations for a multi-skill project scheduling problem with partial preemption

Oliver Polo Mejia, Marie-Christine Anselmet, Christian Artigues, Pierre Lopez

Mixed-integer and constraint programming formulations for a multi-skill project scheduling problem with partial preemption

Oliver POLO MEJIA, Marie-Christine ANSELMET **Christian ARTIGUES, Pierre LOPEZ**

CEA, DEN, DEC, SETC
St Paul lez Durance, France
oliver.polomejia@cea.fr, marie-christine.anselmet@cea.fr

LAAS-CNRS, Université de Toulouse, CNRS
Toulouse, France
lopez@laas.fr, artigues@laas.fr

ABSTRACT: *In this paper, we consider the weekly scheduling problem of activities within one of the research facilities of the French Alternative Energies and Atomic Energy Commission (CEA in short for French). To better represent this problem we propose a new variant of the multi-skill project scheduling problem (MSPSP) involving partial preemption. We describe the new MSPSP variant and we present two formulations for the problem: one using mixed-integer linear programming (MILP) and a second one using constraint programming (CP). Computational experiments on realistic data are carried out and discussed.*

KEYWORDS: *RCPSP, MSPSP, Partial preemption, Scheduling, Nuclear laboratory.*

1 INTRODUCTION

There are not much research works studying the application of optimization techniques for scheduling activities within research facilities. At operational level, scheduling research activities becomes a very complex problem and the literature on this subject is almost non-existent. In this paper we work on the weekly scheduling of the activities within one of the research facilities of the French Alternative Energies and Atomic Energy Commission (CEA in short for French). After analyzing the operations and characteristics of the studied laboratory, we conclude that the problem under consideration amounts to a new extension of the classical Resource-Constrained Project Scheduling Problem (RCPSP).

The RCPSP is a classical scheduling problem that allows the modeling of a broad spectrum of real-life situations. The problem consists in scheduling non-preemptive tasks on limited renewable resources. These tasks are linked together by precedence relationships (task i cannot start until task l is finished). Usually, the objective is to find a solution that minimizes the makespan of the project, while complying both the precedence constraints and the resource constraints.

Formally, the RCPSP can be defined by a 7-tuple (I, d, E, R, B, b, T) where I is a set of activities, d is a vector of activity durations, E is a set of precedence relationships, R is a set of resources, B is a vector of resource availability, b is a matrix of resource demands or consumptions per activity, and T is the set of scheduling periods (Artigues *et al.*, 2013).

Even if the classical version of the RCPSP is very expressive, it cannot cover all the situations that happen in real-life problems. That is why researchers have developed more general or extended versions of the RCPSP using the classical version as starting point. Surveys on this topic are proposed for example by (Hartmann and Briskorn, 2010) and (Orji and Wei, 2013). Among all these variants, we distinguish one that is of great interest for the modeling of the scheduling problem at hand: the Multi-Skill Project Scheduling Problem (MSPSP).

The MSPSP, presented by the first time in (Néron, 2002), combines characteristics of both the classical RCPSP for the project description, and the Multi-Purpose Machine model with the addition of new resource constraints. In this variant a resource is therefore characterized by the set of skills it possesses; and a task is now defined by the number of required resources with a specific competence.

This problem consists in determining a feasible schedule, respecting the precedence constraints between activities and the resource constraints: a resource cannot execute a skill it does not master, cannot be assigned to more than one competence requirement at a given time, and must be assigned to the corresponding activity during its whole processing time. The aim is to minimize the total duration of the project (Bellenguez-Morineau, 2008). Activities in the MSPSP are supposed to be non-preemptive, that means, once started an activity must run continuously until its completeness. The main characteristics of the MSPSP are showed in Figure 1.

With the MSPSP as starting point, adapting some of its characteristics and relaxing the non-preemption constraint, we have developed a new variant that

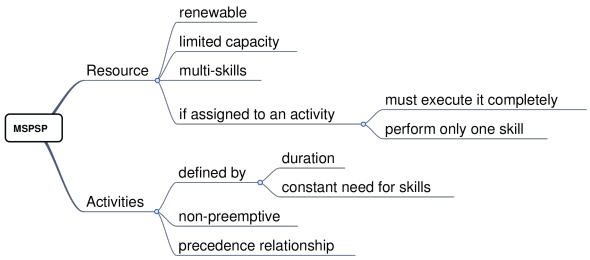

Figure 1 – Characteristics of the MSPSP

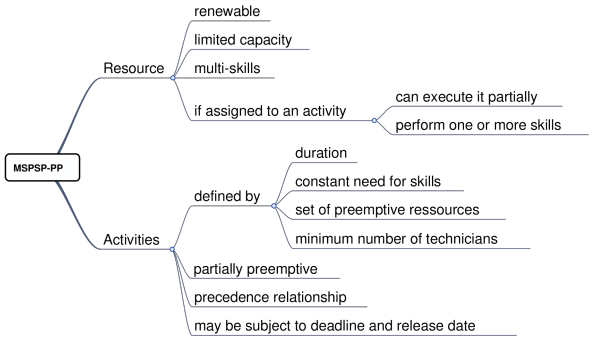

Figure 2 – Characteristics of the MSPSP-PP

should better represent the real-life problem we are trying to solve: an MSPSP with partial preemption.

The remainder of the paper is as follows. In the next section, we describe the problem under consideration. In Section 3, we present the mixed-integer linear programming model representing the partially preemptive MSPSP. As a modeling alternative, a Constraint Programming model is also presented in Section 4. In Section 5, we discuss the computational experiments carried out. Finally, in Section 6 we conclude and discuss future research.

2 PROBLEM DESCRIPTION

When scheduling research or engineering activities, it may be interesting to allow the preemption, in order to reduce the makespan of the project, especially when resource availability is very limited. In our case, due to some safety and operational constraints, proper to nuclear regulation, we can only allow the preemption of a subset of activities. In (Polo Mejia *et al.*, 2017), we proposed several MILP formulations, for a first model of the considered nuclear research facility, where some activities are non-preemptive while some other activities are fully preemptive (but with a penalty every time the activity was preempted). This model only fulfilled partially the operational requirements. Indeed, when working with preemptive scheduling problem, commonly we assume that all resources are released during the preemption periods. However, for some research activities, we are interested in avoiding the release of some equipment or resource having an important setup time.

That is why we propose in this paper to work with a variant allowing the partial release (partial preemption) of resources according to the characteristics of the activities. We must indicate for each activity what resource can be released during the preemption periods. Preemption is now handled in three levels according to the activities characteristics: 1) Non-preemption, for activities where none of the resources can be preempted; 2) Partial preemption, for activities where a subset of resources can be preempted; and 3) Full preemption, for activities where all resources can be preempted.

In order to have a better representation of reality, we need to do some changes over the technicians behavior. Unlike the traditional MSPSP, in our practical case, technicians may respond to more than one skill requirement per activity. Also, due to operational and safety reasons, we need to guarantee a minimum number of technicians present during the execution of the activity. Finally, due to the durations of some activities (larger than technicians' work shifts), we need to relax the constraint stating that the same technician must execute the totality of the activity.

Additionally, we must include some other characteristics to our problem concerning the time windows for scheduling. In the laboratory, the regulatory test must be executed before a restrictive date (deadline). Moreover, some of the activities are in collaboration with other nuclear facilities, such activities are then restricted by a release date fixed by external partners. A recap of our MSPSP with partial preemption (MSPSP-PP) is presented in Figure 2.

The complexity of the MSPSP with partial preemption can be established using the classical RCPSP as 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 (Blazewicz *et al.*, 1983) we can therefore infer that the MSPSP with partial preemption is also strongly NP-hard.

Once defined the characteristics and the complexity of the proposed problem, we proceed to formalize the problem using the two more common approaches in the literature: Mixed-Integer Linear Programming (MILP) and Constraint Programming (CP). These models are presented in the following sections.

3 MILP MODEL

Usually, the variants of the RCPSP and the MSPSP can be modeled using different approaches such as: continuous time-based models based on flows, discrete-time mixed integer linear programming (MILP) formulations, or event-based MILP formulations. An analysis of these approaches allows us to identify the so-called on/off formulation as the most suitable for the preemptive case we are working on. This time-indexed formulation 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.

Using the on/off formulation as basis, we tested two models having as only difference the way in which we modeled the preemption periods. After preliminary computational experiments, one of the models showed significantly better results, and it is presented below:

3.1 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
- $Z_{i,t} \in \{0,1\}$, $Z_{i,t} = 1 \iff$ 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 at time t
- $End_i \in \mathbb{Z}_+$: Completion time of activity i
- $Cmax \in \mathbb{Z}_+$: Project makespan

3.2 Objective function

The most common objective function found in the literature for the RCPSP and the MSPSP is to minimize the project makespan:

$$\min(Cmax) \tag{1}$$

However, as we are working in a research facility, we have the interest to assure that all activities are completed as soon as possible. We can translate this as the minimization of the average completion time for all activities:

$$\min \sum_{i \in I} End_i \tag{2}$$

Using an aggregate objective function may have a negative impact on the solving time of the problem. That is why, we decide to test both objective functions and evaluate the impact of this change.

3.3 Constraints

$$\sum_i O_{j,i,t} \leq DO_{j,t} \ \forall j, \forall t \tag{3}$$

$$\sum_i ((Y_{i,t} + PR_{i,k} * Pp_{i,t}) * Br_{i,k}) \leq DR_{i,k} \ \forall t, \forall k \tag{4}$$

$$(Y_{i,t} + Pc_i * Pp_{i,t}) * Bc_{i,c} \leq \sum_j (O_{j,i,t} * CO_{j,c}) \quad \forall i, \forall t, \forall c \tag{5}$$

$$\sum_j O_{j,i,t} \geq (Y_{i,t} + Pc_i * Pp_{i,t}) * Nt_i \ \forall t, \forall i \tag{6}$$

$$\sum_t Y_{i,t} \geq D_i \ \forall i \tag{7}$$

$$D_l * (1 - Y_{i,t}) \geq \sum_{t'=t}^{T} Y_{l,t'} \ \forall (i,l) \in E, \forall t \tag{8}$$

$$\sum_{t=dl_i+1}^{T} Y_{i,t} \leq 0 \ \forall i \tag{9}$$

$$\sum_{t=1}^{r_i-1} Y_{i,t} \leq 0 \ \forall i \tag{10}$$

$$Pp_{i,t} = Z_{i,t} + W_{i,t} - Y_{i,t} - 1 \ \forall i, \forall t \tag{11}$$

$$Z_{i,t} \geq Y_{i,t'} \ \forall i, \forall t, \forall t' \leq t \tag{12}$$

$$W_{i,t} \geq Y_{i,t'} \ \forall i, \forall t, \forall t' \geq t \tag{13}$$

$$Z_{i,t} \leq \sum_{t'=1}^{t} Y_{i,t'} \ \forall t \tag{14}$$

$$W_{i,t} \leq \sum_{t'=t}^{T} Y_{i,t'} \ \forall t \tag{15}$$

$$End_i \geq t * Y_{i,t} \ \forall t, \forall i \tag{16}$$

$$Cmax \geq End_i \ \forall i \tag{17}$$

Equations (3) ensure that operator's capacities ($DO_{j,t}$) are satisfied. In equations (4), we ensure that all resource requirements ($Br_{i,k}$) are satisfied respecting the resource capacities ($DR_{k,t}$). Parameter $PR_{i,k}$ indicates whether the resource k can be preempted ($PR_{i,k}$=0) or not ($PR_{i,k}$=1). Equations (5) ensure the respect of skill requirements ($Bc_{i,c}$) taking into account the set of skills of technicians ($CO_{j,c}$; $CO_{j,c} = 1$ if technician j has the competence c, 0 otherwise). Parameter Pc_i indicates whether technicians can be preempted (Pc_i=0) or not (Pc_i=1). The constraints given in (6) and (7) ensure the respect of the minimum number of technicians (Nt_i) and duration of activities (D_i), respectively. Precedence constraints are given in (8). Inequalities (9) and (10) are the constraints for deadlines (dl_i) and release dates (r_i). Equations (11) determine whether an activity is preempted or not. Inequalities (12) to (15) are constraints for getting the values of variables

$Z_{i,t}$ and $W_{i,t}$. Equations (16) allows to calculate the completion time of each activity. Finally, inequalities (16) calculate the makespan of the project. Note that it is not necessary to declare variables $Y_{i,t}$, $Z_{i,t}$, End_i and $Cmax$ as integer variables since an integer optimal solution for the other variables enforces integrity on these variables. However, the modern branch-and-bound procedures inside MILP solvers may be able to exploit the knowledge of the integrity of these variables via cuts and preprocessing techniques to converge faster.

4 CP MODEL

In the last decades constraint programming has attracted high attention among experts from many areas of computer science due to its potential for solving hard real life problems. The main idea of constraint programming is to solve problems by stating constraints (requirements) about the problem area and, consequently, finding solutions satisfying all the constraints. The increasing interest for this technique leaded us to use it and evaluate its performance over the studied problem. Our first CP model for the MSPSP with partial preemption is presented below; further research needs to be done in order to improve its performance. We use the IBM CP Optimizer (CPO) as software to model and solve this problem. The presented model refers to the concept of interval variables, a constrained object tailored to scheduling problem, and also to other specific scheduling constraints in CPO.

4.1 Variables

- $itvs_i$: Interval variable between the start and the end of activity i
- $par_{i,p}$: Optional [1] 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

4.2 Objective function

As mentioned before, we want to study the impact of using an aggregate objective function. In our CP model the objective functions are calculated as follows:

Minimize the project makespan:

$$\min(\max_{\forall i \in I} itvs_i.end) \tag{18}$$

[1] Optional interval variables may or may not be present in the solution, so as to satisfy the constraints.

Minimize the average completion time of activities:

$$\min \sum_{i \in I} itvs_i.end \tag{19}$$

4.3 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 (Laborie, 2009). We use this kind of constraint to span the $par_{i,p}$ and $InTech_{j,i,p}$ variable within the $itvs_i$ variables.

Span all the parts p of the activity i within the execution interval of variable i:

$$span(itvs_i, par_{i,p} : \forall p) \ \forall i \tag{20}$$

Span allocation intervals of technicians ($InTech_{j,i,p}$) within the variables $itvs_i$. With constraints (20) this will allow only the necessary technician assignment interval variables to be present in the solution. These constraints stand over the hypotheses that every activity requires at least 1 technician for its processing:

$$span(itvs_i, InTech_{j,i,p} : \forall j, \forall p) \ \forall i \tag{21}$$

noOverlap({b1, .., bn}) constraint: states that none of the interval variables within the set $\{b1, .., bn\}$ overlap over the time. We use this variable to ensure a disjunctive constraint over the technicians:

$$noOverlap(InTech_{j,i,p} : \forall i, \forall p) \ \ \forall j \tag{22}$$

Ensure the activity duration: the *sizeOf* expression is used to access the size (duration) of an interval variable:

$$D_i = \sum_p sizeOf(par_{i,p}) \ \forall i \tag{23}$$

Respect the deadlines and release dates:

$$dl_i \geq itvs_i.end \tag{24}$$

$$r_i \leq itvs_i.start \tag{25}$$

Precedence relationships:

$$itvs_i.end < itvs_l.start \ \forall (i,l) \in E \tag{26}$$

Let us define $rUsage_k$ as a cumulative function indicating the usage of resource k over the time. Also let $pulse(F, h)$ be an elementary pulse function taking the value of h over interval F. We can state the resource constraint as follows:

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

$$rUsage_k <= DR_k \ \forall k \quad (27)$$

For ensuring the skill requirements, we define $SkAll_{i,c}$ as a cumulative function indicating the allocation of skill c for the activity i over the time.

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

The $alwaysIn(F, B, min, max)$ constraint is used to confine the values of a cumulative function F during an interval B inside interval [min,max]. We can then define the skill constraints for preemptive technicians as follows:

$$alwaysIn(SkAll_{i,c}, par_{i,p}, Bc_{i,c}, \infty) \ \forall i : Pc_i = 0, \forall c, \forall p \quad (28)$$

If technicians are declared as non-preemptive the constraint is applied to the interval variable of the activity rather than on its parts:

$$alwaysIn(SkAll_{i,c}, itvs_i, Bc_{i,c}, \infty) \ \forall i : Pc_i = 1, \forall c \quad (29)$$

We must add the constraints on the minimum number of technicians required for executing an activity. In order to do that, we use a cumulative function $AllTech_i$ indicating the number of technicians allocated for an activity over the time. This function is calculated as follows:

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

The constraints are then defined for preemptive technicians as follows:

$$alwaysIn(AllTech_i, par_{i,p}, Nt_i, \infty) \ \forall i : Pc_i = 0, \forall p \quad (30)$$

For non-preemptive technicians we have:

$$alwaysIn(AllTech_i, itvs_i, Nt_i, \infty) \ \forall i : Pc_i = 0 \quad (31)$$

We also need to ensure that there is not overlapping of every activities part variable. However, in order to improve our model we decided not to use the *noOverlap* expression we presented before and state these constraints as follows, so as to break symmetries:

$$par_{i,p}.end < par_{i,s}.start \ \forall i, \forall p, \forall s \in Parts : s > p \quad (32)$$

$$presenceOf(par_{i,p}) \implies presenceOf(par_{i,s}) \quad \forall i, \forall p, \forall s \in Parts : s > p \quad (33)$$

presenceOf() is a boolean constraint, which is true when the interval variable is present. Together with Constraints (20) and (23), these constraints ensure that only the necessary part interval variables are present.

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, F)* that states that whenever the interval variable a is present, it cannot overlap a point t where the step function $F(t) = 0$. We can define these constraints as follows:

$$forbidExtent(InTech_{j,i,p}, PreTech_j) \forall j, \forall i, \forall p \quad (34)$$

Finally, in order to improve the model we added a synchronization constraint between the $InTech_{j,i,p}$ and the $par_{i,p}$ variables for all the activities where technicians can be preempted. The constraint synchronize(a, $\{b_1, ..., b_n\}$) makes intervals $\{b_1, ..., b_n\}$ start and end together with interval variable a (if a is present).

$$synchronize(par_{i,p}, InTech_{j,i,p} : \forall j) \ \forall i : Pc_i = 0, \forall p \quad (35)$$

5 EXPERIMENTAL RESULTS AND DISCUSSION

Computational tests have been carried out using the solver CPLEX 12.7 for the MILP model and CP Optimizer 12.7 for the CP model. We generated our sets of instances using a basic instance generation algorithm that allows the control of certain aspects such as: proportions of preemption type, percentage of activities with deadline and release date, number of precedence relationships, skill number per technicians, etc.

The proposed MILP model requires that the user indicates the initial scheduling horizon (an estimation of project makespan, T) in order to initialize the decision variables. Since we are using a "time-indexed" formulation, this estimation may have an important role in the performance of the model. That is why, the first thing to want to analyze is the behavior of the solving time when the makespan estimation (T) change. We tested the MILP model using the objective function (1) (MILP1). A small set of 7 dummy instances have been generated with a makespan lower than 30 time units (this instance have 20 activities with duration between 1 and 5 time units, 4 precedence relationship, 13 skills, all other characteristics are random).

As shown in Table 1, when the initial estimation gets far from the real value, the time needed to solve the problem increases significantly. This can be explained by the fact that when T increases, the MILP model will need to handle more decision variables to solve to optimality the same instance.

	MILP1 T=30	MILP1 T=50
Instance 1	15.05 sec	60.77 sec
Instance 2	2.05 sec	36.59 sec
Instance 3	15.23 sec	69.93 sec
Instance 4	23.07 sec	138.61 sec
Instance 5	27.23 sec	105.32 sec
Instance 6	36.71 sec	171.65 sec
Instance 7	1403.31 sec	228.98 sec

Table 1 – Time to solve to optimality

	MILP2 T=30	MILP2 T=50
Instance 1	15.00%	36.00%
Instance 2	13.35%	25.68%
Instance 3	13.93%	55.96%
Instance 4	17.85%	58.39%
Instance 5	11.00%	51.42%
Instance 6	13.56%	33.38%
Instance 7	13.84%	50.35%

Table 2 – Gap after 30 min of computation

We did the same test using the objective function (2) (MILP2). Results are shown in Table 2. Since the time required to get the optimal solution is bigger than the fixed time limit, we present the final gap (percentage of deviation between the best feasible found solution and the optimal solution) after 30 minutes of computing. As it can be seen from Table 2, this gap increases with T. This shows us the importance of having a good estimation of the project makespan in order to improve the performance of the proposed MILP model. In the following of this research project, we must develop heuristic methods allowing us to have a good estimation of the project makespan in order to reduce the required time to solve to optimality the instances.

Another observation we can do from the results in Table 1 and Table 2, is that the use of an aggregate objective function has a negative impact in the computation time needed to solve the instances. This may be due to the fact that they can exist a big number of feasible solutions having different values for the aggregate function but with the same makespan. So, when branching the solver cannot cut these branches as fast as it would do it for a makespan optimization. Another possible reason is the quality of upper/lower bounds we can get for an aggregate objective function.

In order to compare the performance of the CP model against the MILP model, we generate a bigger set of 95 heterogeneous instances (15 activities, duration between 1 to 10 time units, 15 skills, 8 resources, 20% of activities with release date and deadline, all other characteristics are random). We solved these instances for the two configurations of the MILP and CP models (CP1 for objective function (18), CP2 for objective function (19)).

		Solved to opt.	Solved not opt.	No able to find a first solution
MILP1	Number	58	32	5
	Mean GAP	-	45.86%	-
	Standard deviation	-	28.93%	-
CP1	Number	1	61	33
	Mean GAP	-	18.63%	-
	Standard deviation	-	23.41%	-

Table 3 – Results configuration 1 (Time lim: 15 min)

Results in Table 3 show that the proposed MILP model outperforms the CP model for configuration 1 (minimization of makespan). The MILP1 was able to solve to optimality a large number of instances (58 out of 95) within the limited time while the CP1 only was able to demonstrate the optimality of 1 instance. Additionally, the number of not solved instances is bigger for the CP. In the following of the project, we must try to improve the performance of the CP model adding or modifying some constraints to break symmetries and reduce the search space.

	Solved to opt.	Solved not opt.	Better value	No first solution
Number (MILP2)	0	78	40	17
Number (CP2)	0	59	19	36

Table 4 – Results configuration 2 (Time lim:15 min)

Table 4 presents the obtained results for configuration 2 (minimization of end times). These results allow us to confirm what we said before, the use of an aggregate objective function has a negative impact in the computation time required to solve the problem (a lower number of optimally solved instances within the limit time). The MILP2 seems to outperform the CP2 being able to get values for a bigger number of instances within the limited time and a lower number of not solved instances. Additionally, MILP2 gets more better solution than the CP2 for the solved instances. Again, we must remember that the proposed CP model is our first version and it should require

some improvements.

A deep analysis of individual results of this test suggested us a difference of performance of the models when the portion of preemptive, semi-preemptive and non-preemptive activities changes. In order to corroborate this, we have generated 4 sets (A, B, C and D) of 40 instances. For each instance in a set, there is a similar instance in the others having as only difference the distribution of the preemption type of activities (this distribution its presented in Table 5). These instances have 15 activities with duration between 1 to 10 time units, 15 skills, 8 resources, 20% of activities with release date and deadline, all other characteristics are random.

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

Table 5 – Distribution of preemption type

We solved the 4 sets of instances to optimality using the configuration 1 of the MILP model. Using a t-test for two paired samples with significance level of 0.05 (Derrick *et al.*, 2017), we try to identify if there is enough statistical evidence to claim a difference of performance (time to solve) according to the distribution of the preemption type. The test results, showed in Table 6, led us to conclude that the MILP model give faster answer when the fraction of preemptive activities increases. For the other kind of preemption, we do not evidence any special impact.

	B	C	D
A	time A < time B p.val= 0.0017	time A < time C p.val=0.0002	Statistically equal p.val=0.0595
B	-	Statistically equal p.val=0.1636	Statistically equal p.val=0.1491
C	-	-	time C > time D p.val=0.0073

Table 6 – T-test for difference in solving time - MILP1

In order to test the impact of the type of preemption over the CP model, we try to solve the same sets of instances using the CP1 configuration. However, the CP1 model was not able to solve to optimality any of the instance within the time limit (30 min). It was not even able to found initial solutions for a large number of instances. This fact made impossible for us to do an analysis similar to the one done for the MILP1. However, we decided to analyze the number of instances for which the model CP1 was able to found at least an initial solution. Results in Table 7 may confirm the same behavior observed for the MILP1: the CP1 model goes faster when it works over highly preemptive instances.

	A	B	C	D
Number of Instances	40	21	7	26

Table 7 – Solved instances per preemption type - CP1

One of the bigger advantages of the MSPSP-PP is the possibility of better handle the partial preemption we can find in real-life problems. When using a traditional preemptive/non-preemptive approach, semi-preemptive activities are usually modeled as non-preemptive. In order to prove the convenience of using a partial preemption approach for reducing the makespan, we decide to analyze the optimal values obtained for the 4 sets of instances.

As expected, the set with most of preemptive activities (set A) allows us to get lower optimal values than the other sets. When comparing the set having a big portion of non-preemptive activities (C) against the one having a big portion of semi-preemptive activities (B), we appreciate that for set B we get lower values than for set C. This confirms the fact that using a semi-preemptive approach may reduce the makespan of the project. With these results we can establish the following relation for makespan value: Preemptive$\leq$Semi-preemptive$\leq$Non-preemptive. This simple relation may help us to calculate the lower/upper bounds we need to develop exact solving algorithms.

The main objective of this project is to schedule the weekly activities of the nuclear laboratory. The first approach is to generate scheduling using the hour as time unit. In a normal week the number of working hours is 108, what give us an idea of the expected makespan. Usually, we must schedule between 40 and 50 activities with a duration between 4 and 20 hours. There is at least 20 technicians, more than 25 resources and at least 20 skills. When trying to solve instances having these characteristics, the models were not able to find optimal solutions after 2 hours of computation, getting an optimality gap going from 20% to 50% for the MILP1. These results lead us to start the development of heuristics methods for solving large instances in reasonable times.

6 CONCLUSIONS

The literature about scheduling research activities is very sparse. That is why, in this paper we presented a new variant of the Multi-skill Project Scheduling Problem (MSPSP) developed to schedule the research activities in a nuclear laboratory: MSPSP with partial preemption. The proposed problem can be easily adapted to a huge number of real-life situations for scheduling activities within very restrictive environments. After describing all the characteristics of the proposed problem, we modeled it using two different techniques: Integer Linear Programming and Constraint Programming.

The first experimental results showed the importance of the initial estimation of the project makespan for the MILP model performance. We also evidenced that using aggregate objective functions has a negative impact on the computation time required to solve the MILP and CP models. Finally, we observed that the MILP model outperformed the CP model for almost all instances and all model configurations, at least for our current CP modeling, which could certainly be improved.

We can conclude that the use of a partially-preemptive approach may lead to reduction of the makespan of the project when compared again a traditional preemptive/non-preemptive approach. Additionally, the proportion of different kinds of preemptions seems to have an impact in the performance of the MILP and CP models. Indeed, the models are faster when the portion of preemptive activities is high.

In practice, the MSPSP with partial preemption may be easily adapted to many real-life situations, especially for scheduling activities within very controlled environments such as pharmaceutical, chemical and aeronautical. For the studied laboratory, the application of the proposed model gives to the engineers the ability to have a more performant schedule in a matter of minutes, at the same time that it eliminates the risk of forgetting constraints, which can happen easily in the manual scheduling process.

As future work, we must study the ways of improvements of the proposed CP model such as adding or modifying some constraints to break symmetries and reduce the search space. We also need to develop heuristics allowing us to have good solutions in reasonable times. The solving times obtained in the experimental tests lead us to develop dedicated algorithms for exact solving. In order to do that we will need to study different approaches for calculating good lower bounds.

REFERENCES

Artigues C., Demassey S. and Néron E., 2013. The Resource-Constrained Project Scheduling Problem. *Resource-constrained project scheduling: models, algorithms, extensions and applications*, John Wiley & Sons.

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

Blazewicz J., Lenstra J.K. and Rinnooy Kan A.H.G., 1983. Scheduling projects subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, 5, pp. 11-24.

Derrick B.,Toher D and White P., 2017. How to compare the means of two samples that include paired observations and independent observations. *Tutorials in Quantitative Methods for Psychology, Vol 13, Iss 2, Pp 120-126 (2017)*, 13(2), pp. 120-126.

Hartmann S. and Briskorn D., 2010. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 207(1), pp. 1-14.

Laborie P., 2009. IBM ILOG CP Optimizer for detailed scheduling illustrated on three problems. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, Springer, Berlin, Heidelberg, pp. 148-162.

Montoya C., Bellenguez-Morineau O., Pinson E. and Rivreau D., 2015. Integrated column generation and lagrangian relaxation approach for the multi-skill project scheduling problem. *Handbook on Project Management and Scheduling*, Springer International, pp. 565-586.

Néron E., 2002. Lower bounds for the multi-skill project scheduling problem. *Proceedings of the Eighth International Workshop on Project Management and Scheduling*, pp. 274-277, Valencia, Spain.

Orji I. M. J. and Wei S., 2013. Project scheduling under resource constraints: A recent survey. *International Journal of Engineering Research and Technology*, 2(2), pp. 1-20.

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