A Constraint Programming Model for the Scheduling Problem with Flexible Maintenance under Human Resource Constraints

Meriem Touat[1,2], Belaid Benhamou[3] and Fatima Benbouzid-Si Tayeb[2]

[1]*Ecole Supérieure des Sciences Appliquées Alger (ESSA-Alger), BP 474 Place des Martyrs, Alger 16001, Algeria*

[2]*Laboratoire des Méthodes de Conception des Systèmes (LMCS), Ecole Nationale Supérieure d'Informatique (ESI), BP 68M 16309 Oued-Smar Alger, Algeria*

[3]*Laboratoire LSIS, Domaine Universitaire de Saint-Jérôme, Batiment Polytech, Université Aix-Marseille, Avenue Escadrille Normandie-Niemen, 13397 Marseille Cedex 20, France*

Keywords: Single Machine Scheduling, Flexible Maintenance Planning, OPL, Constraints Programming, Human Resource Constraints.

Abstract: In this work, we tackle the scheduling problem that considers both production and flexible preventive maintenance on a single machine where the human resource constraints (the availability and the competence) are taken into account. We propose a mathematical formulation for the problem that is expressed in the constraint programming (CP) paradigm as a set of constraints. This CP modeling had been implemented using Ilog CP Optimizer. Experiments were first carried out on small instances to compare our CP implementation with that one carried out in Mixed Integer Linear Program programming (MILP) presented in (Touat et al., 2021), then the CP implementation had been tested on large instances and encouraging results were obtained.

1 INTRODUCTION

In this work, we deal with the single machine scheduling problem with unavailability intervals due to the preventive maintenance activities that could be periodic or flexible. In theory, this kind of problems have been proved to be NP-hard (Lee and Liman, 1992). To be solved, both exact and heuristic methods have been proposed in the literature. Indeed, several mathematical modeling have been developed according to the constraints taken into account ((Chen, 2008), (Cui and Lu, 2014), (Liu et al., 2015), (Yang et al., 2011), (Mashkani and Moslehi, 2016)). However, given the high computational complexity of the problem, exact algorithms could solve only small instances in practice. Therefore, specific heuristics and meta-heuristics have been extensively used to solve larger instances of this problem ((Chen, 2008), (Low et al., 2010), (Luo et al., 2015), (Yang et al., 2011), (Zammori et al., 2014), (Yazdani et al., 2017)).

A new and more realistic single machine problem is introduced in works (Touat et al., 2017), (Touat et al., 2018) and (Touat et al., 2021). In order to reflect the reality of production shops, the authors assumed that maintenance activities must be done by human resources, characterized by a competence levels and qualifications allowing them or not to execute the maintenance activities with different durations. In addition, these human resources are not available permanently, but in specified intervals that assess the feasibility or not of the resulting schedules. First, a genetic algorithm is proposed in (Touat et al., 2017) to solve the problem in the uncertain context. Further, both an exact method expressed as a Mixed Integer Linear Program (MILP) and a metaheuristic are proposed in (Touat et al., 2021) to deal with, respectively, small and large instances of the problem. The exact method is solved using the Ilog IBM Cplex, while the metaheuristic is inspired from the Guided Local Search method (GLS).

In addition to exact methods that are based on integer programming (IP) and mixed integer linear programming (MILP) formulations, the Constraint Programming paradigm (CP) framework had been widely applied to solve the scheduling problems. The commercial solver IBM Ilog CP Optimizer that is used to solve CP formulations, is a very known system in the literature (Laborie et al., 2018). This solver is dedicated to find optimal solutions, and when it is not able to produce a such solution, it produces a good quality solution in a reasonable amount of time. Furthermore, the OPL (Optimization Programming Lan-

Touat, M., Benhamou, B. and Tayeb, F.
A Constraint Programming Model for the Scheduling Problem with Flexible Maintenance under Human Resource Constraints.
DOI: 10.5220/0010800700003116
In *Proceedings of the 14th International Conference on Agents and Artificial Intelligence (ICAART 2022) - Volume 3*, pages 195-202
ISBN: 978-989-758-547-0; ISSN: 2184-433X

guage) make easier the scheduling problems modeling. For more details on this subject, the reader could refer to (Laborie et al., 2018).

Several research works using CP framework to solve scheduling problems have been proposed. In (Laborie, 2009), the authors used OPL to solve the CP model of three problems that are the flow-shop with earliness and tardiness costs, the satellite scheduling and the personal task scheduling. A flexible job shop scheduling problem that incorporates machine operators to minimize the makespan is studied in (Kress and Müller, 2019). In (Mokhtarzadeh et al., 2020), authors developed a CP approach to solve the problem known as human-robot collaboration proper to allocation of tasks to humans and robots to minimize makespan. Authors of (Hauder et al., 2020) introduced a new resource-constrained project scheduling problem (RCPSP) where both decisions (activity flexibility and time flexibility) are integrated to minimize the makespan and maximize both the balanced length of selected activities (time balance) and the balanced resource utilization (resource balance). In (Lunardi et al., 2020), authors considered the on line printing shop scheduling problem that can be seen as a flexible job shop scheduling problem with sequence flexibility and precedence constraints to minimize the makespan. A simultaneous scheduling of production and material transfer in a job shop environment to minimize the makespan is presented in (Ham, 2020).

In other scheduling context, authors of (Ornek et al., 2020) proposed a scheduling based constraint programming modeling to solve the flight-gate assignment problems. In (Polo-Mejía et al., 2019), authors propose a way to apply operation research techniques to a particular CP modeling to schedule research activity within a nuclear facility. The use of scheduling models represent an improvement of the facility safety and also allows researchers to save time. In (Qin et al., 2020), authors formulated the container terminals of seaports as a special hybrid flow shop scheduling problem. The authors of (J. Kinable and Smith, 2021) propose a CP formulation for a Snow Plow Routing Problem (SPRP) which involves finding a set of vehicle routes for a street network service in a pre-defined area, while accounting for various vehicle constraints and traffic restrictions.

In this work, we deal with the single machine scheduling problem and flexible maintenance planning where each maintenance activity is assigned to a human resource characterized by a competence and some availability intervals. We aim to optimize both production and maintenance criteria. Since the MILP modeling and the exact method proposed in (Touat et al., 2021) suffer from scalability issues, we propose here a new mathematical modeling and an exact resolution method that are based on the constraint programming (CP) paradigm to solve efficiently this new scheduling problem. We expressed the CP modeling in the OPL language and implemented it in IBM Ilog CP Optimizer. The CP model has been tested on a large number of instances of this problem and the results obtained have been compared to those obtained with the MILP model (Touat et al., 2021).

In the rest of the paper, we will describe the studied problem in Section 2 and its modeling in Section 3, then, the experimental results and comparison with the MILP method in Section 4. Section 5 concludes the work and gives some research perspectives.

2 THE SINGLE-MACHINE SCHEDULING PROBLEM DESCRIPTION

In this section, we describe the considered scheduling problem. First, we give the notations, then introduce the constraints and the objective functions.

2.1 The Used Variables and Notations

Let $J = J_1, J_2, ...J_N$ be a set of N jobs to be processed by a single machine. Each job J_i requires a given known deterministic and non-negative processing time p_i and should be completed before a due date d_i. The job J_i starts at t_i and finishes at c_i.

Besides, preventive maintenance must be undertaken in order to maintain a high availability of the machine. In this work, we consider a single flexible maintenance M with multiple occurrences. The occurrences are encoded by M_i, $i \in \{N+1, ..N+Nb\_Occ\}$ with a duration p' and follows a given period T^*. A maintenance M_i should be completed within a time window $TI_i = [Tmin_i, Tmax_i]$ representing its tolerance interval. We assume that the first time-window is arranged in advance. The starting time and the completion time of M_i are denoted respectively by t_i and c_i.

M_i require a human resource to be treated. Indeed, the maintenance service is composed of R human resources *(HR)*. Each human resource HR_r $(r = 1..R)$ is characterized by a competence level $Comp_r$ allowing to execute a maintenance task with a duration ph_r. Moreover, each resource HR_r has a timetabling which determines its availability expressed by specifying for each resource HR_r a set $AI_r = \{AI_{rl} : l = 1..m\}$ of m availability intervals (AI). More precisely, $AI_r = \{[LB_{r1}, UB_{r1}], .., [LB_{rm}, UB_{rm}]\}$ where LB_{rl} and UB_{rl}

denote respectively, the lower and upper bounds of the lth availability interval $(l = 1..m)$ of AI_{rl} $(l = 1..m)$.

2.2 The Problem Constraints

The constraints of the problem are as following:

- All jobs J_i, $\forall j \in \{1,...,N\}$ are independent and available for processing at time 0 and each job is either waiting for processing or being processed by the machine at any given time.
- A job is processed once and only once on the machine.
- There is no preemption, i.e., a production job or a maintenance activity is not interrupted after it has started.
- The machine can only process one job at given a time, and all the jobs are non-resumable, i.e. a preventive maintenance activity must be planed either before or after a job which cannot be disrupted.
- The machine is stopped at least one time to perform maintenance interventions, i.e. the machine is not able to process all production jobs without being maintained at least once.
- The i^{th} tolerance interval depends on the completion time of the $(i-1)^{th}$ maintenance occurrence M_{i-1}.
- We do not perform any maintenance operation after the processing of the last job, since we seek scheduling over a production horizon.
- A maintenance M_i must be treated by only one human resource and in only one availability interval.
- A human resource can only treat one maintenance activity at a given time.
- The effective maintenance activity duration ph_i varies according to the human resource HR_r charged to execute M_i. $ph_i = Comp_r \times p'$.

2.3 The Optimization Criteria

The production objective f_p is to find a permutation of N production jobs that minimizes the sum of tardiness T_i, when the schedule also includes maintenance activities (Eq. 1).

$$\begin{cases} f_p = \sum_{i=1}^{N} T_i \\ T_i = max(0, c_i - d_i) \quad , \quad i = 1..N \end{cases} \tag{1}$$

The maintenance objective f_m consists in minimizing the sum of earliness/tardiness of all the maintenance operations with respect to the pre-specified maintenance intervals $TI_i = [Tmin_i, Tmax_i]$. It is achieved when the maintenance activity is more profitable and before the equipment loses its optimum performance. It could be planed before $Tmin_i$, and considered in advance and its earliness is expressed by the variable E_i. It could also be planed after $Tmax_i$ and in this case, it is considered as late, and its tardiness is represented by the variable T_i. The maintenance operations are planned by taking into account the human resource constraints (Eq. 2).

$$\begin{cases} f_m = \sum_{i=N+1}^{N+Nb\_Occ} (E_i + T_i) \\ E_i = max(0, Tmin_i - t_i) \qquad i = N+1..N+Nb\_Occ \\ T_i = max(0, c_i - Tmax_i) \qquad i = N+1..N+Nb\_Occ \end{cases} \tag{2}$$

To optimize both production and maintenance criteria, we try to minimize the global function defined as follows (Eq. 3):

$$\begin{cases} f = \alpha \times f_p + \beta \times f_m \\ \alpha + \beta = 1 \end{cases} \tag{3}$$

According to the notation proposed in (Touat et al., 2021) based on the classification given in (Graham et al., 1979), one could denote the considered problem by $1/d_i, M, TI_i, g(i,r), AI_r/\alpha \sum T_i + \beta \sum (E_i + T_i)$. It is then easy to see that the addressed problem is *NP-hard* in the strong sense since the simplified version of this problem family denoted $N/1/d_i/\sum T_i$ has been demonstrated to be *NP-hard* (Kan, 1976).

3 CP MODELING OF THE CONSIDERED SINGLE-MACHINE SCHEDULING PROBLEM

Here, we introduce the CP modeling of the problem which is based on OPL. We recall that a first MILP model has already been proposed in (Touat et al., 2021). The CP model uses some variables that we define in Table 1.
The variable declaration phase written under OPL is given below:

The declaration phase in OPL:
1: *dvar interval prod[i in 1..N] size p[i];*
2: *dvar interval maint[i in 1..Nb_Occ];*
3: *dvar interval intermaint[i in 1..Nb_Occ] size Tmax-Tmin;*
4: *dvar interval maintOp[i in 1..Nb_Occ][j in 1..R] optional size Comp[j];*
5: *stepFunction Breaks[k in 1..R][i in 1..m] = stepwise{100->0; 0->LB[k][i]; 100->UB[k][i];*

0};
6: *dexpr int tardinessP = sum(i in 1..N) maxl(0, endOf(prod[i])-d[i]);*
7: *dexpr int tardinessM = sum(j in 1..Nb_Occ) maxl(0,endOf(maint[j])-endOf(intermaint[j]));*
8: *dexpr int earlinessM = sum(j in 1..Nb_Occ) maxl(0, startOf (intermaint[j])-startOf(maint[j]));*

Table 1: CP model variables.

Variables and their definition
interval prod[N]: An array of N interval variables corresponding to N production jobs which will contain the scheduling of production jobs
interval maint[Nb_Occ**]:** An array of interval variables for the Nb_Occ maintenance activities which will contain the scheduling of maintenance activities
interval intermaint[Nb_Occ**]:** An array of interval variables for the Nb_Occ tolerance intervals which will contain the tolerance intervals of maintenance activities
interval maintOp[Nb_Occ**][R]:** An array of interval variables for the Nb_Occ maintenance activities and the R human resources that will contain scheduling of maintenance occurrences according to the assigned human resource
stepFunction Breaks[R][m]: An array of a defined step function variables for the R human resources and m maintenance, that determines for each human resource their availabilities

In lines 1 and 2, we declare both the intervals of production jobs J_i with the precision of their processing times p_i and the intervals of maintenance activities. Line 3 defines the tolerance intervals for the maintenance activities. Line 4 gives the scheduling interval of each maintenance activity M_i according to the human resource charging to execute M_i. Line 5 defines a step function Breaks that represents the unavailability of human resource HR_R according to a maintenance activity. In Ilog CP Optimizer, step functions are constant structures of the model that are represented by a set of steps associated with a value. The value of the step function is 0% when a human resource k is available. That is the value of the function is 0% on the time windows $AI_r = \{[LB_{r1}, UB_{r1}], .., [LB_{rm}, UB_{rm}]\}$ and 100% between these time windows. Lines 6, 7 and 8 show the expressions of the objective functions representing the production tardiness, the maintenance tardiness and the earliness.

The resulting CP model of the considered problem is given bellow:

Minimize:

$$f_{CP} = \alpha \times tardinessP + \beta \times (earlinessM + tardinessM) \tag{4}$$

Subject to:

$$\begin{gathered} startOf(intermaint[1]) == Tmin; \\ endOf(intermaint[1]) == Tmax; \end{gathered} \tag{5}$$

$$\begin{gathered} startOf(intermaint[i]) == endOf(maint[i-1]) + T^*, \\ i = 2..Nb\_Occ \\ endOf(intermaint[i]) == endOf(maint[i-1]) + T^* \\ +(Tmax_1 - Tmin_1), i = 2..Nb\_Occ \end{gathered} \tag{6}$$

$$\begin{gathered} alternative(maint[i], all(jin1..R)maintOp[i][j]) \\ i = 1..Nb\_Occ \end{gathered} \tag{7}$$

$$\begin{gathered} forbidExtent(maintOp[i][j], Breaks[j][i]) \\ j = 1..R, i = 1..Nb\_Occ \end{gathered} \tag{8}$$

$$noOverlap(append(prod, maint)) \tag{9}$$

The global objective function (Eq. 4) is used to minimize the addition of the sum of tardiness of production jobs and the sum of earliness/tardiness of the maintenance activities. The constraints 5 and 6 compute the tolerance interval of each maintenance activity. We recall that for M_1, both $Tmin$ and $Tmax$ are given. Constraint 7 ensures that each maintenance activity is treated by only one human resource and in only one availability interval. Indeed, for each maintenance activity its scheduling interval must be selected from the list of the possible intervals. Constraint 8 meets the requirement that there is no overlapping between the maintenance intervals and the human resource unavailability intervals. Indeed, the constraint is used to indicate that a given interval variable cannot overlap a particular date from the list of possible scheduling maintOp and the stepwise function Breaks that determine the availabilities. Finally, Constraint 9 guarantees that prod and maint intervals do not overlap.

4 EXPERIMENTS

To validate our CP model, we implemented it in the Optimization Programming Language (OPL) and used the IBM Ilog Cplex Optimization Studio Community 20.1 version to solve instances of the considered problem that are expressed with respect to our CP model. The Tests are carried on a personal computer with an *Intel Core i7 2.70 GHz* CPU and *16 Gb* RAM memory under *Windows* operating system.

We used the same method as the one given in (Touat et al., 2021) to generate the data (the problem instances). Indeed, we used two types of data. The first one is related to production and maintenance data, while the second one is related to the human resource part. We present, the generated data in Table 2.

Table 2: Data generation.

Variables	***Values***
N	$N \in \{9,10,11,12,13,15,18,20,40,60\}$
p_i, d_i	Randomly generated
p'	The average of p_i
T^*	150
$[Tmin, Tmax] = \triangle T^*$	$\triangle T^* = 0.05 \times T^*$
HR	Two human resources
$Comp_i$	distributed as $U]0,2[$
ph_i	$ph_i = Comp_i \times p'$
AI_l	Strict Availability Interval (SAI) and Large Availability Interval (LAI)

We performed tests on two subsets of benchmarks. First we deal with small ones varying according to their sizes N (the number of production jobs) from *9* to *13* jobs. We generated ten instances of each size and obtain an amount of 50 instances of production/maintenance. Secondly, we considered large size instances for the CP model ranging from *15* to *60* jobs. We also generated ten instances for each given size, then got 50 large instances of production/maintenance. In total we tested 100 instances of the considered problem.

We considered, for each instance of production/maintenance four classes of human resources according to the human characteristics. The different classes correspond to higher and lower human resource competences (denoted by, respectively, LC and HC) and to strict and large availability intervals (denoted by, respectively, SAI and LAI). It results that four classes of experiments (SAI/LC, SAI/HC, LAI/LC and LAI/HC) are performed.

In order to tackle the problem in the simplified way, the values of both control parameters α and β of Equation 3 are set to 0.5 for all the considered tests.

4.1 Results of the CP Model on Small Size Instances

We aim here to evaluate the CP model on small size instances and compare it to the MILP model proposed in (Touat et al., 2021) which were implemented and tested on the same resource as the current work. The restriction to small size instances was dictated by the fact that the MILP model cannot handle large instances. Since the MILP is able to find the optimal solution in a time limit fixed to one hour up to only 13 jobs, we compared the results obtained from the CP model on the small size instances $N \in \{9,10,11,12,13\}$ and we considered for each instance all of the four different classes of tests. We then performed an amount of 4X10X5=200 tests for the small size benchmark family.

The relative percentage deviation *(RPD)* is used as an index to evaluate the solution quality and the performance of the proposed CP model. This index is given by the following equation:

$$RPD = \frac{(f_{CP} - f_{MILP})}{f_{MILP}} \times 100 \tag{10}$$

where f_{MILP} is the average global objective function of the MILP modeling and f_{CP} is the average global objective function of the CP modeling (Eq. 4).

The obtained results are shown in Table 3. For each instance, we report the average value of the global objective function obtained by running the 10 instances for all of the four classes of tests, the elapsed time (T(s)), the percentage of instances for which the proposed CP find the optimal solution and the RPD values.

Table 3: The results of the proposed CP model and the ones of the MILP model on small instances.

N	***SAI/LC***					
	MILP		***CP***			***RPD***
	f_{MILP}	*T(s)*	f_{CP}	*T(s)*	*Best*	
9	423.20	3.39	435.35	2.02	90%	2.87%
10	402.75	2.87	410.90	2.18	80%	2.02%
11	439.85	12.39	445.65	2.18	90%	1.32%
12	503.20	197.73	508.40	2.80	90%	1.03%
13	416.75	553.67	427.30	2.79	90%	2.53%
AVG					88%	1.95%
N	***SAI/HC***					
	MILP		***CP***			***RPD***
	f_{MILP}	*T(s)*	f_{CP}	*T(s)*	*Best*	
9	135.05	3.34	141.25	2.14	90%	4.59%
10	147.55	3.37	155.50	2.15	70%	5.39%
11	174.85	5.98	179.15	2.57	90%	2.46%
12	216.00	18.15	219.85	3.38	80%	1.67%
13	183.45	84.56	183.45	2.37	100%	0%
AVG					86%	2.82%
N	***LAI/LC***					
	MILP		***CP***			***RPD***
	f_{MILP}	*T(s)*	f_{CP}	*T(s)*	*Best*	
9	304.50	10.02	304.50	2.61	100%	0%
10	284.80	17.95	289.45	2.52	70%	1.63%
11	355.95	61.10	362.05	3.17	80%	1.71%
12	366.85	553.67	385.30	4.51	60%	5.03%
13	348.55	634.32	353.45	3.88	50%	1.40%
AVG					72%	2.44%
N	***LAI/HC***					
	MILP		***CP***			***RPD***
	f_{MILP}	*T(s)*	f_{CP}	*T(s)*	*Best*	
9	106.80	3.93	106.80	2.08	100%	0%
10	121.65	3.26	122.10	2.57	90%	0.37%
11	147.35	7.82	151.25	2.65	70%	2.65%
12	176.40	18.56	182.60	3.15	60%	3.51%
13	172.40	159.80	172.40	2.83	100%	0%
AVG					70%	2.17%

The obtained results show that for all the tested instances, the RPD does not exceed 5.40. We can even remark that the CP model find the best solution in 100% for four cases of the tests that are respectively Benchmark 9 (LAI/LC and LAI/HC classes) and Benchmark 13 (SAI/HC and LAI/HC classes). In average, the RPD of each class of tests varies between 1.85 and 2.82. Furthermore, the best solution is found in more than 80% when the intervals are strict and more than 70% when the intervals are large.

According to elapsed times, the CP model improves the performance of the MILP model, especially on instances where the size is between 11 and 13. We can see that CP model spent between 2 s and 5 s to find the solution for all of the classes and all benchmarks. It outperforms drastically the MILP model. However, the MILP spent a considerable time to find the optimal solution for the benchmarks having a size ranking from 11 to 13, in comparison to CP model.

Another point to address, is the scalability of the models. That is the number of variables (NV) and constraints (NC) involved in both models to encode each instance. We present in Table 4 the average of these data for each benchmark. For more analysis precision, we present in Table 5 for the CP model, the number of variables and constraints for all of the ten instances of Benchmark 9.

Table 4: The number of variables and constraints of both the CP and MILP models on small instances.

N	***MILP***		***CP***	
	NV	*NC*	*NV*	*NC*
9	180	151	19	17
10	186	171	20	16
11	239	215	22	18
12	272	252	25	21
13	255	269	24	18

We remark from Table 4 that both the number of variables and constraints of the MILP are larger than the ones of the CP model. This reason contributes in the sense of obtaining elapsed times for the CP model lower than those of the MILP models. We recall that within a time limit set at one hour, the MILP model could not resolve instances with a size greater than 13. The difference in number of variables (NV) and number of constraints (NC) of the MILP model is of the order of nine times the measurements in number of variables and constraints of the CP model. In addition, this gap tends to increase when the number of variables increases, thus making the CP model more advantageous for solving large instances.

Table 5: The number of variables and constraints of the CP model of Benchmark 9.

Instance	***VN***	***CN***
1	22	21
2	18	15
3	22	21
4	22	21
5	14	9
6	22	21
7	18	15
8	22	21
9	22	21
10	14	9

Now, if we are interested in the particular case of Benchmark 9 of the CP model given in Table 5, we can notice that the number of variables varies between 14 and 22 and the number of constraints varies between 9 and 21. Generally the number of the variables and constraints of an instance of this problem depend on its number of production tasks and the number of its maintenance tasks. As the number of production tasks is fixed at 9 for the ten instances of Benchmark 9, we understand that the variation of the number of variables *NV* and that one of the constraints *NC* comes from the fact that the number of maintenance activities changes.

4.2 Results of the CP Model on Large Size Instances

As we mentioned previously, we extended the experiments to larger size instances. It should be noted that the free version of IBM Ilog Cplex reaches its calculation limits when the size of the benchmark is equals to 60 jobs. Indeed, Ilog CP Optimizer Community Edition solves problems with search spaces up to 2^{1000}. We consider here the benchmarks having the size $N \in \{15, 18, 20, 40, 60\}$. We generated ten instances for each of the four classes of each benchmark. In total, we experimented 5X10X4=200 tests. First, we considered Benchmark 20 that we experimented in order to fix the execution time limits to one hour. We remarked that for the majority of the generated instances (80% of the instances) for this benchmark, the best solutions is found in less than one hour. Then for the rest of instances, we fixed the time limit to one hour.

The obtained results are shown in Table 6. For each benchmark and class of tests, we give the average values of all of the production, the maintenance, the global objective functions and the elapsed times (T(s)) spent to solve the considered instance.

First of all, we can see in Table 6 that with the CP model we manage to solve instances of this problem containing up to 60 production tasks in one hour of elapsed time while the MILP model fails to resolve instances of this problem having more than 13 production

Table 6: The CP results for the large size problems.

N	***SAI/LC***			
	f_p	f_m	f	$T(s)$
15	2316.90	269.00	1292.95	16.67
18	2919.3	347.8	1633.55	304.86
20	4275.80	361.70	2318.75	3309.60
40	16205.50	755.90	8490.75	3600
60	27275.60	1173.10	14224.35	3600
N	***SAI/HC***			
	f_p	f_m	f	$T(s)$
15	683.50	44.90	364.20	10.80
18	784.3	72.3	428.3	18.292
20	1246.40	109.60	678.00	445.05
40	5420.20	214.30	2816.75	3600
60	7353.70	363.20	3873.30	3600
N	***LAI/LC***			
	f_p	f_m	f	$T(s)$
15	1629	146	887.5	67.086
18	2644.7	235.3	1439	1048.638
20	3591.80	282.90	1936.35	2741.16
40	13766.20	539.70	7155.40	3600
60	25020.50	852.30	12936.40	3600
N	***LAI/HC***			
	f_p	f_m	f	$T(s)$
15	591.4	26.50	308.95	13.675
18	714.4	27.4	370.9	16.61
20	1061.50	50.20	555.85	577.24
40	4943.20	117.00	2542.25	3600
60	6947.00	191.30	3569.15	3600

jobs under the same elapsed time condition.

We can note that the order of magnitude of the elapsed time changes from a few seconds for the instances having 15 production tasks to a few minutes for the instances having 18 production tasks then changes approximately to one hour time for the instances having more than 20 production tasks. This proves the high impact of availability constraints even when the number of production tasks does not increase significantly. But the tendency remains that the complexity varies exponentially depending on the number of production tasks that represent the size of the problem.

We can see that, for all benchmarks, the quality of the three objective functions f_p,f_m, and f is better when considering human resource with high competence (*HC*) for both classes of instances *SAI* and *LAI*. More precisely, we got the rank LAI/HC, SAI/HC, LAI/LC and SAI/LC for the different classes. The main reason of this result is due to the fact that a maintenance operation executed by a human resource with a high competence and large availability intervals has more possible insertion sites and consequently it leads to more opportunities to optimize the objective function. Indeed, with a high competence, the maintenance duration decreases and could be inserted inside its tolerance interval since the human resource has large availability intervals.

In summary, we can say that the CP model represents a considerable improvement for the MILP model. We recall that the CP model finds the optimal solution in almost 80% of cases and when the latter is not reached it returns a good solution very close to the optimal. This model then represents a very good compromise between the exact MILP model and the non-exact methods called metaheuristics.

5 CONCLUSION

In this paper, we deal with the scheduling problem of production and maintenance activities that considers the competence and availability human resource constraints.

The main contribution of this paper is the introduction of a CP model to formulate the studied scheduling problem. This CP model had been implemented in OPL language and presents a good alternative to the MILP modeling proposed in (Touat et al., 2021).

To validate our CP model, we used it to express several instances of the problem that we solved with the IBM Ilog Cplex software engine and compared our approach to the MILP model (Touat et al., 2021) on small instances. The obtained results show that the proposed CP model is powerful and outperforms the MILP model. Also, it is a good compromise between exact methods like the the MILP approach and metaheuristics, since it usually succeeds to find the optimal solutions in approximately 79% of the checked small instances in a very low elapsed time.

Further, the CP model was used to represent and solve relatively large instances of the studied problem. We were able to solve instances with up to 60 production tasks in one hour of elapsed time while the MILP model cannot resolve instances of more than 13 production tasks in one hour of lab time.

For future work and perspectives, we first plan to study a multi-objective version of the problem and compare it to the version introduced in this work, then we want to generalize our study for versions of the problem with several machines.

REFERENCES

Chen, J.-S. (2008). Scheduling of nonresumable jobs and flexible maintenance activities on a single machine to minimize makespan. *The International Journal of Advanced Manufacturing Technology*, 190:90–102.

Cui, W.-W. and Lu, Z. (2014). Integrated production scheduling and periodic maintenances on a single machine with release dates. *IEEE International Confer-*

ence on Automation Science and Engineering (CASE), 197:Taipei, Taiwan.

Graham, R.-L., Lawler, E.-L., Lenstra, J.-K., and Kan, A.-R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete mathematics*, 5:287–326.

Ham, A. (2020). ransfer-robot task scheduling in job shop. *International Journal of Production Research*, DOI: 10.1080/00207543.2019.1709671.

Hauder, V.-A., Beham, A., Raggl, S., Parraghb, S.-N., and Affenzeller, M. (2020). Resource-constrained multi-project scheduling with activity and time fexibility. *IComputers and Industrial Engineering*.

J. Kinable, W.-J. v. H. and Smith, S.-F. (2021). Snow plow route optimization: A constraint programming approach. *IISE Transactions*, 53:6:685–703.

Kan, A.-R. (1976). *Machine scheduling problems: classification, complexity and computations*. Springer.

Kress, D. and Müller, D. (2019). Shop scheduling problem with machine operator constraints. *IFAC PapersOnLine*, pages 94–99.

Laborie, P. (2009). Ibm ilog cp optimizer for detailed scheduling illustrated on three problems. *Lecture Notes in Computer Science*, https://doi.org/10.1007/978-3-642-01929-6-12.

Laborie, P., Rogerie, J., Shaw, P., and Vilim, P. (2018). Ibm ilog cp optimizer for scheduling: 20+ years of scheduling with constraints at ibm/ilog. *Constraints*, https://doi.org/10.1007/s10601-018-9281-x.

Lee, C.-Y. and Liman, S.-D. (1992). Single machine flow-time scheduling with scheduled maintenance. *Acta Informatica*, 29:375–382.

Liu, M., Wang, S., Chu, C., and Chu, F. (2015). An improved exact algorithm for single-machine scheduling to minimise the number of tardy jobs with periodic maintenance. *International Journal of Production Research*, DOI:10.1080/00207543.2015.1108535.

Low, C., Ji, M., Hsu, C.-J., and Su, C.-T. (2010). Minimizing the makespan in a single machine scheduling problems with flexible and periodic maintenance. *Applied Mathematical Modelling*, 34:334–342.

Lunardi, W.-T., Birgin, E.-G., Laborie, P., Ronconi, D.-P., and Voos, H. (2020). Mixed integer linear programming and constraint programming models for the online printing shop scheduling problem. *Optimization and Control (math.OC)*, DOI: 10.1016/j.cor.2020.105020.

Luo, A., Cheng, T.-C.-E., and Ji, M. (2015). Single-machine scheduling with a variable maintenance activity wenchang. *Computers and Industrial Engineering*, 79:168–174.

Mashkani, O. and Moslehi, G. (2016). Minimising the total completion time in a single machine scheduling problem under bimodal flexible periodic availability constraints. *International Journal of Computer Integrated Manufacturing*, 29:323–341.

Mokhtarzadeh, M., Tavakkoli-Moghaddam, R., Vahedi-Nouri, B., and Farsi, A. (2020). Scheduling of human-robot collaboration in assembly of printed circuit boards: a constraint programming approach. *International Journal of Computer Integrated Manufacturing*, DOI: 10.1080/0951192X.2020.1736713.

Ornek, M.-A., Ozturk, C., and Sugut, I. (2020). Integer and constraint programming model formulations for flight-gate assignment problem. *Operational Research*, https://doi.org/10.1007/s12351-020-00563-9.

Polo-Mejía, O., Artigues, C., Lopez, P., and Basini, V. (2019). Mixed-integer/linear and constraint programming approaches for activity scheduling in a nuclear research facility. *International Journal of Production Research*, DOI:10.1080/00207543.2019.1693654.

Qin, T., Du, Y., Chen, J.-H., and Sha, M. (2020). Combining mixed integer programming and constraint programming to solve the integrate d scheduling problem of container handling operations of a single vessel. *European Journal of Operational Research*, 285:884–901.

Touat, M., Benbouzid, F., and Benhamou, B. (2021). Exact and metaheuristic approaches for the single-machine scheduling problem with flexible maintenance under human resource constraints. *International Journal of Manufacturing Research*.

Touat, M., Bouzidi-Hassini, S., Benbouzid-Sitayeb, F., and Benhamou, B. (2017). A hybridization of genetic algorithms and fuzzy logic for the single-machine scheduling with flexible maintenance problem under human resource constraints. *Applied Soft Computing*, 59:556–573.

Touat, M., Tayeb, F. B.-S., and Benhamou, B. (2018). An effective heuristic for the single-machine scheduling problem with flexible maintenance under human resource constraints. *Procedia Computer Science*, 126:1395–1404.

Yang, S.-L., Maa, Y., Xu, D.-L., and Yang, J.-B. (2011). Minimizing total completion time on a single machine with a flexible maintenance activity. *Computers and Operations Research*, 38:755–770.

Yazdani, M., Khalili, S.-M., Babagolzadeh, M., and Jolai, F. (2017). A single-machine scheduling problem with multiple unavailability constraints: A mathematical model and an enhanced variable neighborhood search approach. *Journal of Computational Design and Engineering*, 4:46–59.

Zammori, F., Braglia, M., and Castellano, D. (2014). Harmony search algorithm for single-machine scheduling problem with planned maintenance. *Computers I& Industrial Engineering*, 76:333–346.