MIP models for a two-machine open shop problem and a server with set-up times

1st Nadia Babou
Laboratoire RECITS, Faculté de mathématiques
Université des Sciences et de la Technologie Houari Boumédiène
Alger, Algérie
nadiababou00@gmail.com

2nd Mourad Boudhar
Laboratoire RECITS, Faculté de mathématiques
Université des Sciences et de la Technologie Houari Boumédiène
Alger, Algérie
mboudhar@yahoo.fr

3th Djamal Rebaine
Département d'informatique et de mathématique
Université du Québec à Chicoutimi
Saguenay (Québec), Canada
Djamal_Rebaine@uqac.ca

***Abstract*—In this paper, we are addressing the two-machine open shop problem with a single sever. A job must be prepared by the server and then will directly be processed on a machine. We show the $\mathcal{NP}$-completeness in the strong sense of a restricted case. Then, we propose two mixed integer programming formulations to solve the general problem. Finally, we analyze their performance by an experimental study we conducted.**

***Index Terms*—Open shop, set-up times, server, makespan, MIP model**

I. Introduction

In scheduling theory, it is generally assumed that the set-up time necessary for processing a job is either negligible or part of the processing time. This assumption does not remain valid if the set-up depends on additional resources of limited capacity. In this case, a set-up cannot be started if the resources are occupied even though a machine and a job are available. This will cause idle times that can significantly affect the quality of the resulting solution. Therefore, the set-up has to be considered as a distinguished part from the processing.

The type of set-up times we are considering in this paper can be seen as the time during which we prepare the necessary resources (machines, tools, etc.) to process a job. The preparation can be the loading and positioning of a job on its machine, or settings that must be done on the machine with the presence of the job on it. Thus the set-up depends on the job to be processed and require its availability. Moreover they are handled by resources, other than the machines, that are of limited capacity. The additional resources can be robots, servers, operators that are only needed during the set-up phase.

Before proceeding and since we are addressing a variant of the open shop, let us describe the standard two-machine open shop scheduling problem. Given are two machines M_1 and M_2 to process a set $T = \{T_1, T_2, \ldots, T_n\}$ of n jobs available at time $t = 0$. The processing of job T_i, $i = 1, \ldots, n$, on machine M_j, $j = 1, 2$, is called operation, namely O_{ji}. Furthermore, the processing order of a job on the machines is arbitrary and has to be chosen during the process of constructing a solution. We seek a schedule that minimizes the overall completion time, also called the makespan. This problem is solvable in linear time by either the algorithm of Sahni and Gonzalez [1] or Pinedo and Schrage [2].

In the variant of the precedent problem, we are addressing in this paper, an additional resource, as discussed above, is available in one unit *i.e.* a server. The server can only handle one set-up at a time.

The paper is organized as follows. Section II is devoted to the literature review. In section III we present our problem and in section IV we give its complexity. In section V, we present the two mixed integer programming formulations we developed and the experimental study we conducted to analyze their performance. We concluded the paper in section VI.

II. Literature review

Many works had considered scheduling problems under resource constraints for setting the jobs. In what follows we are giving a literature review of those works.

In [3]–[6] is treated the problem with identical parallel machines and a server. Papers [7]–[12] have considered the problem with two identical parallel machines and a server. [13] has treated the problem with m identical parallel machines and $(m-1)$ servers. The same problem has been treated in [14] but with k servers ($k < m$). When [15] has studied the problem with two dedicated parallel machines and a server.

The authors of [16] have considered the problem with two identical parallel machines and k resources of unit demand to set-up the jobs. Paper [17] approached the two-machine flow shop problem and a server with setting and removal times. In [15], the two-machine flow shop and the no-wait two-machine flow shop with a server are treated. Also [18] has studied the flow shop problem with a server.

Let us mention in the last position, with more details, the works that addressed similar problem to the problem we are studying. First, [15] treated a two-machine open shop problem with set-up times. The set-ups here are handled by a single server. In this problem a job can be prepared even if its processing on another machine is not yet completed. They showed that the problem is $\mathcal{NP}$-hard in the strong sense. Also in [19] is studied a two-machine open shop but with a set of additional resources of limited capacity. A job may require a subset of the resources to be setted up. A feasible schedule is a schedule where the set-up of a job cannot start if this job is still being processed on the other machine.

III. Definition of the problem

In the problem we are studying, we add to the standard open shop problem with two machines, described above, a server $S1$. In this case a job must be prepared by the server then will be processed without preemption on machine M_j. The period during which the server sets up job T_i for its processing on M_j is called the set-up time, denoted hereafter $s_{ji} \geq 0$ and the processing period is called processing time, denoted hereafter $p_{ji} \geq 0$, $i = 1, \ldots, n$; $j = 1, 2$. The server will be available again at the end of a set-up phase and can instantly start setting up another job on the second machine, when processing continues on the first machine.

This makes it possible to distinguish in the execution of a job two separate phases that succeed each other. The first phase is a set-up phase followed immediately by a processing phase. It should be noted that the set-up of a job cannot start on one machine if it is being processed on the other machine. The set-up cannot be done if the job is not available. The objectif is to minimize the overall completion time. The problem is denoted, $O2, S1||C_{max}$.

We point out the differences between our problem and the two similar others mentioned above. The work [19] has treated the problem with a set of additional resources. Each resource has a certain capacity. When in our problem, there is only one additional resource, a single server. In [15] is, only, proved the strong $\mathcal{NP}$-hardness of the problem with one additional resource. A set-up can be carried out in advance even if the job is being processed on another machine. While in the problem we are addressing a job must be available to be set up.

Example 1: For the sake of clarity, let us consider the instance, given in Table I.

If the Pinedo and Scharage algorithm [2] is applied to solve it, we obtain the two solutions illustrated in Fig. 1. Both solutions are the scheduling of the same sequence of jobs, but the unavailability of the server gives us two possibilities, start on M_1 or M_2.

T_i	T_1	T_2	T_3
s_{1i}	2	3	1
p_{1i}	2	3	1
s_{2i}	2	2	1
p_{2i}	4	1	3

TABLE I: Example

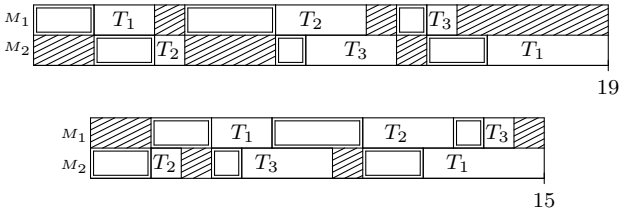

Fig. 1: The two obtained solutions

We notice that the makespan of the two solutions are different. The optimal solution is given in Fig. 2. Its optimality can be verified by enumerating all the possible solutions.

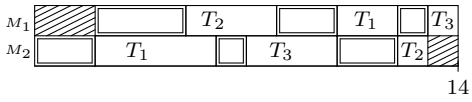

Fig. 2: Optimal solution

IV. Complexity of the problem

In this section, we show the strong $\mathcal{NP}$-completeness of the case where the set-up time of a job on each machine is equal to its processing time, denoted $O2, S1|p_{1i} = s_{1i}, p_{2i} = s_{2i}|C_{max}$.

The problem we used in the reduction process is the 3-partition problem known to be $\mathcal{NP}$-hard in the strong sense [20]. It is defined as follows:

***3-partition**:* Given a set $A = \{a_1, \ldots, a_{3n}\}$ of $3n$ positive integers , with $\sum_{i \in N} a_i = nB$, for a certain integer B, with $B/4 < a_i < B/2$, $i \in N = \{1, \ldots, 3n\}$. Does there exist a partition of N into n subsets N_j, $j = 1, \ldots, n$ such that $\sum_{i \in N_j} a_i = B$?

Theorem 1: $O2, S1|p_{1i} = s_{1i}, p_{2i} = s_{2i}|C_{\max}$ is $\mathcal{NP}$-hard in the strong sense.

Proof 1: Given an instance of 3-partition, we construct the following instance I of $O2, S1|p_{1i} = s_{1i}, p_{2i} = s_{2i}|C_{\max}$ with $4n + 1$ jobs. We denote the partition jobs as P-jobs. T'-jobs are n identical jobs, $T_k, k = 3n + 1, \ldots, 4n$, other than the P-jobs. The set-up and the processing times are summarized in Table II.

	P-jobs: T_i, $i = 1, \ldots, 3n$	T'-jobs T_{3n+k}, $k = 1, \ldots, n$	T_{4n+1}
s_{1i}	0	$3B$	$B/2$
p_{1i}	0	$3B$	$B/2$
s_{2i}	a_i	B	0
p_{2i}	a_i	B	0

TABLE II: Set-up and processing times

It is clear instance I can be constructed in polynomial time. In the following we show that the corresponding decision

problem of $O2, S1|p_{1i} = s_{1i}, p_{2i} = s_{2i}|C_{\max}$ has a solution S with $C_{\max}(S) \leq y = (6n+1)B$ if, and only if, 3-partition problem has a solution.

Let us first assume that 3-partition has a solution, and $A_\ell, \ell = 1, \ldots, n$, are the required subsets. Let $N_\ell, \ell = 1, \ldots, n$, be the corresponding subsets of $N = \{1, \ldots, 3n\}$ according to the partition. The schedule is illustrated in Fig. 3 and is as follows.
The processing on M_1 is without idle time, according to the following order:

$$(T_{3n+3}, T_{3n+4}, \ldots, T_{4n}, T_{3n+1}, T_{3n+2}, T_{4n+1}).$$

The total idle time on M_2 is equal to $(2n+1)B$ time units. The processing on this machine starts at time $3B$, according to following order:

$$(T_i, i \in N_1, T_{3n+1}, T_i, i \in N_2, T_{3n+2}, \ldots, T_i, i \in N_{n-1}, T_{4n-1}, T_i, i \in N_n, T_{4n}).$$

An idle time of $2B$ time units occurs between the processing of a job $T_{3n+\ell}$ and jobs $T_i, i \in N_\ell, \ell = 1, \ldots, n$.

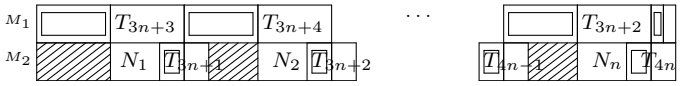

Fig. 3: Structure of the schedule

Let us now assume that the scheduling problem has a solution with a makespan equal to $y = (6n+1)B$, we show that 3-partition has a solution. The schedule has the following structure.

There is no idle time on M_1 whatsoever, since its total execution time is equal to the makespan. The processing starts on M_1 otherwise the first set-up on M_2 will cause an idle time on this machine.

To avoid idle times on M_1, caused by the unavailability of the server, we have to schedule the set-ups on M_2 in parallel with the processing phases on M_1. The set-up times of T'-jobs are all equal to B. The only processing times larger than B are the processing of T'-jobs on M_1. Therefore, the set-up of each T'-job on M_2 has to be scheduled in parallel with the processing of another T'-job on M_1.

The total idle time on M_2 is equal to $(2n+1)B$ time units. Which is the difference between the completion time and the overall processing time on M_2. Any supplementary idle time on this machine will increase the value of the makespan.

Since the processing starts on M_1 then, the first set-up on this machine will cause an idle time on M_2 of $3B$ time units. No job can be started on M_2 while the server sets up jobs on M_1. To prevent this time interval from being idle, a processing phase of a job must be scheduled at the same time as the start of these set-ups. The largest processing times on M_2 are the processing of the T'-jobs. If the processing phase of a T'-job on M_2 starts at the same time as a set-up phase of another T'-job on M_1 then we obtain the smallest total idle time possible on M_2. This idle time is unavoidable and equals $3B+2(n-1)B$ time units. The partial structure of the schedule is as pictured by Fig. 4.

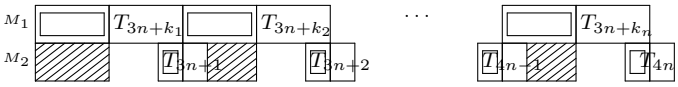

Fig. 4: Partial structure of the schedule

The only job left to schedule on M_1 is T_{4n+1}. There are two possibilities: schedule it in either the first or the last position. If we schedule it in the first position then the makespan will exceed y. Therefore, T_{4n+1} has to be scheduled in the last position.

The jobs left to schedule are the P-jobs on M_2. In order to not exceed the value of the makespan, these jobs must be scheduled during the free time intervals on M_2. There are n free time intervals of the same length, $2B$ time units.

Assume P_ℓ are the P-jobs scheduled during the time interval ℓ, $\ell = 1, \ldots, n$. In order to have a schedule with a makespan of value y, the following equation must be verified:

$$\textstyle\sum_{i \in P_\ell}(s_{2i} + p_{2i}) = 2B, \ell = 1, \ldots, n.$$

Then, $\quad \textstyle\sum_{i \in P_\ell} 2a_i = 2B, \ell = 1, \ldots, n.$

Therefore, $\quad \textstyle\sum_{i \in P_\ell} a_i = B, \ell = 1, \ldots, n.$

If we define subsets $N_\ell = P_\ell$, then $N_\ell, \ell = 1, \ldots, n$ represent a solution of 3-partition.

V. MIP MODELS

Since the problem we are addressing is $\mathcal{NP}$-hard in the strong sense, the integer programming approach is well justified. In this section, we propose two mixed integer programming formulations.

A. First Model

This first model is based on directly defining the starting time of each operation and the precedence relations between operations. Most of the constraints are based on defining the precedence relation between the operations. Thus, the decision variables of the model are:
t_{ji} : starting time of operation j; $j = 1, 2$ of job T_i ; $i = 1, \ldots, n$.
The binary decision variables that express the precedence relations between operations

$$x_{jij'i'} = \begin{cases} 1 & \text{if } O_{ji} \text{ starts before } O_{j'i'}; \\ 0 & \text{otherwise} \end{cases}$$
$$j, j' = 1, 2; \;\; i, i' = 1, \ldots, n$$

Let us also define, for $j = 1, 2$ and $i = 1, \ldots, n$, the following:

$$a_{ji} = \begin{cases} 1 & \text{if } s_{ji} > 0 \\ 0 & \text{otherwise} \end{cases}$$

M: an upper bound of the makespan.
The objectif is to minimize the total completion time denote y. The model is the following :

(P1)

$$\min y$$

$$s.t. \quad a_{j'i'} \times (t_{ji} + s_{ji}) \leq t_{j'i'} + M \times x_{jij'i'};$$
$$i, i' = 1, \ldots, n;\ j, j' = 1, 2;$$
$$(i, j) \neq (i', j'). \quad (1)$$
$$a_{ji} \times (t_{j'i'} + s_{j'i'}) \leq t_{ji} + M \times (1 - x_{jij'i'});$$
$$i, i' = 1, \ldots, n;$$
$$j, j' = 1, 2;\ (i, j) \neq (i', j'). \quad (2)$$
$$t_{ji} + s_{ji} + p_{ji} \leq t_{ji'} + M \times x_{jiji'};$$
$$i, i' = 1, \ldots, n;$$
$$j = 1, 2;\ i \neq i'. \quad (3)$$
$$t_{ji'} + s_{ji'} + p_{ji'} \leq t_{ji} + M \times (1 - x_{jiji'});$$
$$i, i' = 1, \ldots, n;$$
$$j = 1, 2;\ i \neq i'. \quad (4)$$
$$t_{ji} + s_{ji} + p_{ji} \leq t_{j'i} + M \times x_{jij'i};$$
$$i = 1, \ldots, n;$$
$$j, j' = 1, 2;\ j \neq j'. \quad (5)$$
$$t_{ji} + s_{ji} + p_{ji} \leq y;$$
$$i = 1, \ldots, n;\ j, j' = 1, 2. \quad (6)$$

- (1) and (2) ensure that the server sets up only one job at a time.
- (3) and (4) ensure that a machine processes only one job at a time.
- (5) ensures that operations of the same job will not be processed at the same time.
- (6) indicates that the completion time of all jobs are smaller or equal to y.

B. Second Model

Since the set-ups cannot run in parallel, then a set-up order is defined for the operations. This results in a permutation of operations. The model is summarized in (P2).

The decision variables are the following for $i = 1, \ldots, n$; $j = 1, 2$; $k = 1, \ldots, 2n$:

$x_{jik} = \begin{cases} 1 & \text{if operation } O_{ji} \text{ is in position } k \\ 0 & \text{otherwise} \end{cases}$

t_k : starting time of the k^{th} set-up, $k = 1, \ldots, 2n$.

M: an upper bound.

(P2)

$$\min y$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \sum_{j=1}^{2} x_{jik} = 1;\ k = 1, \ldots, 2n. \quad (1)$$
$$\sum_{k=1}^{2n} x_{jik} = 1;\ j = 1, 2;\ i = 1, \ldots, n. \quad (2)$$
$$t_k + \sum_{i=1}^{n} \sum_{j=1}^{2} x_{jik} \times s_{ji} \leq t_{k+1};$$
$$k = 1, \ldots, 2n - 1. \quad (3)$$
$$t_k + \sum_{i=1}^{n} x_{jik} \times (s_{ji} + p_{ji}) - t_{k'} \leq M \times$$
$$(1 - \sum_{i=1}^{n} x_{jik'});\ j = 1, 2;$$
$$k = 1, \ldots, 2n - 1;\ k' = k + 1, \ldots, 2n. \quad (4)$$
$$t_k + (s_{ji} + p_{ji}) - t_{k'} \leq M \times (2 - x_{jik} -$$
$$x_{j'ik'});\ i = 1, \ldots, n;\ j \neq j' = 1, 2;$$
$$k = 1, \ldots, 2n - 1;\ k' = k + 1, \ldots, 2n. \quad (5)$$
$$t_k + \sum_{i=1}^{n} \sum_{j=1}^{2} x_{jik} \times (s_{ji} + p_{ji}) \leq y$$
$$k = 1, \ldots, 2n. \quad (6)$$

- (1) ensures that there is only one operation on a position.
- (2) makes that each operation takes one position.
- (3) ensures that two successive set-ups dot not overlap.
- (4) ensures that a machine processes one job at a time.
- (5) imposes that the operations of the same job are not processed simultaneously.
- (6) establishes that the completion time of all jobs are smaller or equal to y.

C. Comparing the models

We first compare the components of the two models, the number of variables and the number of constraints. The size of the two models are summarized in Table III. As we can see, the number of binary variables in the second model is larger than in the first model. The number of integer variables are the same for both models. The second difference resides in the number of constraints where the second model has more constraints than the first model.

Model	Binary variables	Integer variables	Constraints
(P1)	$2n^2$	$2n + 1$	$8n^2 - n$
(P2)	$4n^2$	$2n + 1$	$12n^3 - 12n^2 + 11n - 1$

TABLE III: Size of the mathematical models

Let us now compare the performance of the two above models. We conducted a computational experiment on an IBM ILOG CPLEX Optimization Studio 12.7. The number of jobs varies in $\{5, 10, 15, \ldots, 75\}$. The set-up times and the processing times are generated randomly, according to the variation of a certain parameter $c \in \{0.0, 0.1, 0.2, \ldots, 0.9\}$. The total processing times $P_{ji} = p_{ji} + s_{ji}$ and the set-up times of each operation are generated from a uniform distribution in $[\min\{10, n\}, \max\{10, n\}]$ and $[\lfloor c \times P_{ji} \rfloor, \lfloor (0.3 + c) \times P_{ji} \rfloor]$, respectively. For a value of the two parameters (n, c) we generate 10 instances. A run time limit of 3600 seconds is imposed for each instance. In what follows we compare the performance of the two models by comparing the average run time given in seconds.

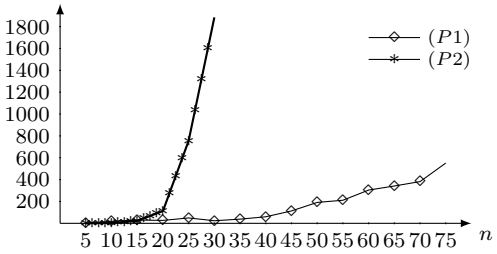

Fig. 5: $c = 0.0$

For the first type, $c = 0.0$, as we can see in Fig. 5, model (P1) outperforms model (P2) and it can solve instances of up to 75 jobs. Whereas (P2) run time increases rapidly to the point where it can solve only instances of at most 30 jobs.

Concerning the second type, $c = 0.1$, model (P1) is still the outperforming model but it is less efficient than it was for the first type. It is able to solve instances of up to 65 jobs. However, even though (P2) is faster for this type than it was for the previous one, it can solve instances of at most 30 jobs as illustrated in Fig. 6.

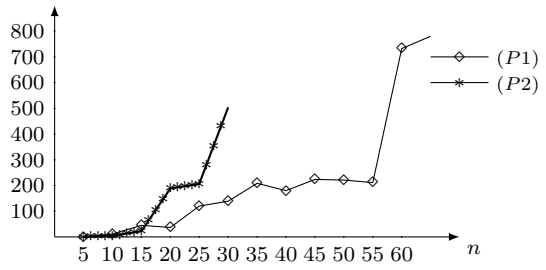

Fig. 6: $c = 0.1$

For the third type, $c = 0.2$, as shown in Fig. 7, the two models have the same performance quality. Both are able to solve instances for a number of jobs up to 30.

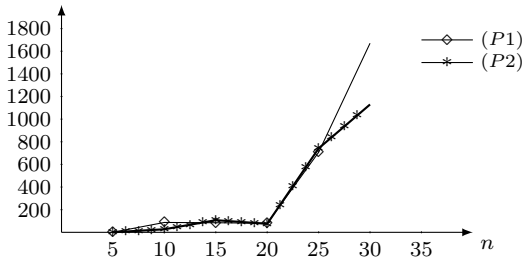

Fig. 7: $c = 0.2$

When it comes to the fourth type, $c = 0.3$, we can clearly see that it constitutes the hardest type instances. Model (P2) can solve more instances but only those with up to 15 jobs. Model (P1) could not solve any instance for $n = 10$. However it was able to solve instances with 15 and 20 jobs. The running times of (P1) and the performance of (P2) for the first interval are summarized in Table IV and Fig. 8, respectively.

n										
5	0.78	0.67	0.62	0.71	0.71	0.85	0.89	0.57	0.87	0.67
10	-	-	-	-	-	-	-	-	-	-
15	208.29	-	-	-	284.37	-	607.49	933.07	-	-
20	291.82	-	669.59	-	109.69	-	-	-	-	-

TABLE IV: Performance of (P1) for $c = 0.3$

For $c = 0.4$, both models were able to solve the same size instances. However, model (P2) is faster than model (P1), which makes it the outperforming model for this type. As we can see in Fig. 9, both models solve instances up to 25 jobs. Clearly model (P2) is more efficient than model (P1) for instances with $c = 0.4$.

Regarding instances with $c = 0.5$ and $c = 0.6$, the performance of (P2) is better as instances up to 40 jobs may

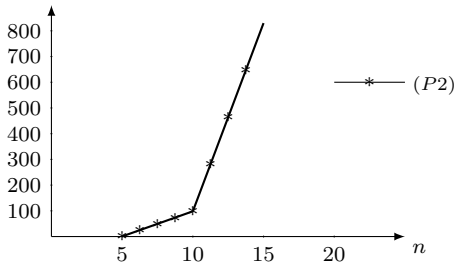

Fig. 8: $c = 0.3$

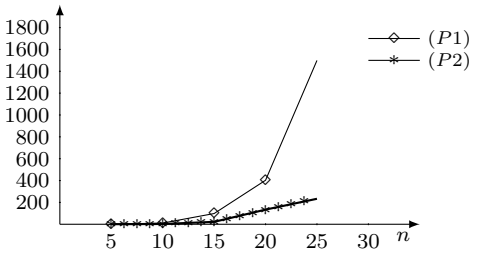

Fig. 9: $c = 0.4$

be solved, whereas (P1) is much slower and was only able to solve instances of at most 25 jobs. The results for these types are summarized in Fig. 10 and 11.

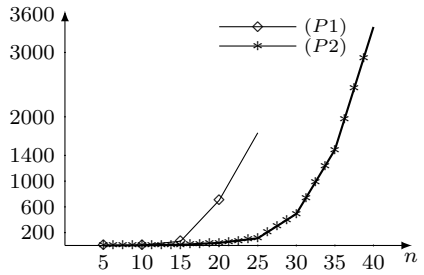

Fig. 10: $c = 0.5$

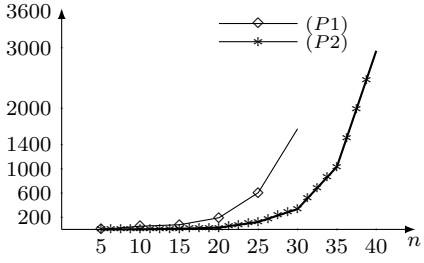

Fig. 11: $c = 0.6$

For the last three types, $c = 0.7, 0.8$ and 0.9, model (P2) remains the outperforming model solving instances of up to 40 jobs, even if the performance of (P1) improved for these types compared to last four ones. Model (P1) is only able to solve instances up to 35, as illustrated in Fig. 12, 13 and 14.

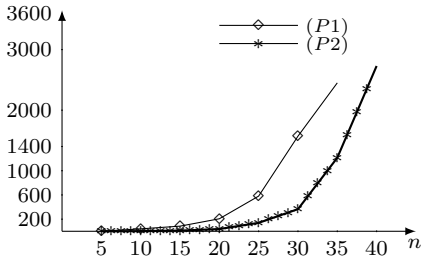

Fig. 12: $c = 0.7$

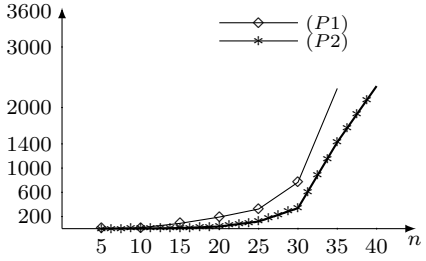

Fig. 13: $c = 0.8$

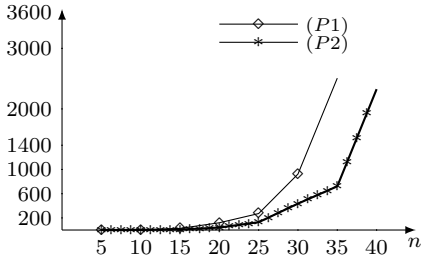

Fig. 14: $c = 0.9$

It is easy to distinguish the types of instances for which model (P1) outperforms model (P2). Indeed, model (P1) has a better performance for the type of instances with $c < 0.2$. Both models share the same performance for instances with $c = 0.2$. For $c > 0.2$, model (P2) becomes the outperforming model. Nevertheless, we may conclude that both models are efficient.

VI. Conclusion

We have studied in this paper the two-machine open shop problem with the presence of a server that sets up the jobs before their processing. We showed the $\mathcal{NP}$-hardness in the strong sense of a restricted case. Then, we proposed two mixed integer linear programming formulations to solve the general problem. Both models are shown, through an extensive numerical experiment, to be useful in the sense that one is better than the other for different instances types. The two models complete each other, since one is effective when the other is not and vice versa.

References

[1] T. Gonzalez and S. Sahni, "Open shop scheduling to minimize finish time", Journal of the ACM, 23, 665-679, 1976.

[2] M. Pinedo and L. Schrage, "Stochastic shop scheduling: A survey", in Dempster MA. (Ed.) Deterministic and stochastic scheduling, Proceedings of the NATO Advanced Study and Research Institute on Theoretical Approaches to Scheduling Problems, Durham, England, pp. 81-96, 1981.

[3] S. A. Kravchenko and F. Werner, "Parallel machine scheduling problems with a single server", Mathematical and Computer Modelling, 26, pp. 1-11, 1997 .

[4] S. A. Kravchenko and F. Werner, "A heuristic algorithm for minimizing mean flow time with unit setups", Information Processing Letters, 79, pp. 291-296, 2001.

[5] P. Brucker , C. Dhaenens-Flipo , S. Knust , S. A. Kravchenko and F. Werner, "Complexity results for parallel machine problems with a single server",Journal of Scheduling, 5, pp. 429-457, 2002.

[6] M. Y. Kim and Y.H. Lee, "MIP models and hybrid algorithm for minimizing the makespan of parallel machines scheduling problem with a single server", Computers and Operations Research, 39, pp. 2457-2468, 2012.

[7] C. P. Koulamas, "Scheduling two parallel semiautomatic machines to minimize machine interference", Computers and Operations Research, 23, pp. 945-956, 1996.

[8] N. G. Hall, C.N. Potts and C. Sriskandarajah, "Parallel machine scheduling with a common server", Discrete Applied Mathematics, 102, pp. 223-243, 2000.

[9] A. H. Abdekhodaee and A. Wirth, "Scheduling parallel machines with a single server: Some solvable cases and heuristics", Computers and Operations Research, 29, pp. 295-315, 2002.

[10] A. H. Abdekhodaee, A. Wirth and H.S. Gan, "Equal processing and equal setup time cases of scheduling parallel machines with a single server", Computers and Operations Research, 31, pp. 1867-1889, 2004.

[11] A. H. Abdekhodaee, A. Wirth and H.S. Gan, "Scheduling two parallel machines with a single server: The general case", Computers and Operations Research, 33, pp. 994-1009, 2006.

[12] K. Hasani, S.A. Krevenchenko and F. Werner, "Scheduling Two Parallel Machines with a Single Server to Minimize Forced Idle Time", International Journal of Production Research, 52, pp. 4-24, 2014.

[13] S. A. Kravchenko and F. Werner, "Scheduling on parallel machines with a single and multiple servers", Otto-von-Guericke-Universitat Magdeburg, Preprint, 30/98, pp. 1-18,1998.

[14] F. Werner and S. A. Kravchenko, "Scheduling with Multiple Servers",Automation and Remote Control, Vol. 71(10), pp. 2109-2121, 2010.

[15] C. A. Glass, Y. M. Shafransky and V. A. Strusevich, "Scheduling for Parallel Dedicated Machines with a Single Server", Naval Research Logistics, 47, pp. 304-328, 2000.

[16] W. Labbi , M. Boudhar and A. Oulamara, "Scheduling two identical parallel machines with preparation constraints", International Journal of Production Research, 55(6), pp. 1531-1548, 2017.

[17] T. C. E. Cheng, G. Wang and C. Sriskandarajah, "One operator - two-machine flowshop scheduling with setup and dismounting times", Computers and Operations Research, 26, pp. 715-730, 1999.

[18] P. Brucker, S. Knust and G. Wang, "Complexity results for flow-shop problems with a single server", European Journal of Operational Research, 165, pp. 398-407, 2005.

[19] A. Oulamara, D. Rebaine and M. Serairi, "Scheduling the two-machine open shop problem under resource constraints for setting the jobs", Annals of Operations Research, 356 , pp. 211-333, 2013.

[20] M. R. Garey and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", A Series of Books in the Mathematical Sciences. (Ed.): Victor Klee, San Francisco, Calif.: W. H. Freeman and Co. pp. x+338. ISBN 0-7167-1045-5, 1979.