Bilevel modeling and solving Flexible Job Shop Scheduling Problem

No Author Given

No Institute Given

Abstract.

Job shop scheduling problems (JSSP) are among the most intensive combinatorial problems studied in literature. The flexible job shop problem (FJSP) is a generalization of the classical JSSP where each operation can be processed by more than one resource. The FJSP problem covers two difficulties, namely, machine assignment problem and operation sequencing problem. The objective of this research is to propose a new bilevel mathematical model to solve the FJSP. The considered objective function is to minimize makespan (C_{max}). To the best of our knowledge, no other exact method was presented till date where the FJSP variant was solved with bilevel scheme. Our bilevel model proved its effectiveness according to the models existing in the literature, and returned less computational time and better objective function according to the classical monolevel MILP model, and several approximate approaches from the literature.

Keywords: Flexible Job shop Scheduling Problem, Bilevel Optimization, Mathematical Modeling.

1 Introduction

;

The most complex scheduling problem studied in the literature is the Job Shop scheduling Problem (JSP). The JSP is a branch of the industrial production scheduling problems. It is among the hardest combinatorial optimization problems [Garey et al., 1976]. The FJSP is an extension of the classical JSP. It consists in assigning operations to one machine out of a set of alternative machines. These operations have to be processed on an available machine in a specified order. In addition to the operation scheduling problem, the FJSP presents an additional difficulty which is the operation assignment problem where each operation is assigned to one of alternative machines. Hence, the FJSP is more computationally difficult than the JSP. Due to its complexity, a very limited number of papers exists in the literature dealing with exact methods. The branch and bound algorithm is one of the traditional exact methods used for the FJSP. The performance of this technique depends on the instance and the initial upper values. [Saidi-Mehrabad and Fattahi, 2007] used a branch and bound algorithm for the FJSP. They obtained optimal solutions for small size instances in a reasonable time. Furthermore, mathematical modeling was well used to solve the FJSP [Özgüven et al., 2010]. we mention [Behnke and Geiger, 2012] who used CPLEX software to solve FJSP using

constraint programming. [Saidi-Mehrabad and Fattahi, 2007] proposed a Mixed Integer Linear Programming (MILP) model to solve the FJSP. A set of 20 instances of small and medium size was solved with MILP using LINGO software. [Özgüven et al., 2010] introduced a MILP model for the FJSP. Their model was tested on 20 instances using CPLEX optimizer. The time spent to solve the 10 small size instances and the objective function were better than those obtained in [Saidi-Mehrabad and Fattahi, 2007]. Furthermore, they obtained optimal solutions for five of the medium size instances and better bounds for the five other instances, while the solutions obtained in [Saidi-Mehrabad and Fattahi, 2007] were not optimal for any of those instances. [Mousakhani, 2013] proposed a MILP model to solve FJSP with sequence-dependent setup times (SDST) to minimize the total tardiness objective function. MILP techniques models are very useful to solve FJSP. But, they may be unfeasible for large size scheduling problems due to their complexity. The FJSP problem covers two difficulties, namely, machine assignment problem and operation sequencing problem. Motivated by this structure, we propose a bilevel MILP model to solve the problem.

Bilevel optimization is a special kind of multilevel optimization where one problem is embedded within another. The outer optimization task is the upper-level optimization task, and the inner optimization task the lower-level optimization task. These problems involve two kinds of variables, namely, the upper-level variables and the lower-level variables. [von Stackelberg, 1934] is the first researcher who formulated a bilevel hierarchical game composed of two players. An increasing attention was given to these problems in the recent years due to its number of practical applications and the advantages provided when using bilevel optimization techniques. The first formulation of bi-level programming was proposed in 1973 by [Bracken and McGill, 1973]. After that in 1977, Candler and Norton are the first authors who use the designation of bi-level and multi-level programming [Candler and Norton, 1977].

A bi-level optimization problem (BOP) is a specific problem with two levels of tasks in which decision variables are split into two groups that are controlled by two decision makers called leader (on the upper level) and follower (on the lower level). Both decision makers have an objective function of their own and a set of constraints on their variables. Furthermore, there are coupling constraints that connect the decision variables of leader and follower. The nested structure of the overall problem requires that a solution to the upper level problem may be feasible only if it is an optimal solution to the lower level problem. A generic bi-level optimization problem can be formulated as follows: A BOP can be stated formally as follows:

$$\underset{x_u \in X_U, x_l \in X_L}{Min} F(x_u, x_l) \tag{1}$$
$$s.t. \begin{cases} x_l \in ArgMin \{f(x_u, x_l), \} \, g_i(x_u, x_l) \leq 0, \\ G_j(x_u, x_l) \leq 0, \; i = 1, ..., I \; and \; j = 1, ..., J \end{cases}$$

where x_u represents the upper level decision vector and x_l represents the lower level decision vector. The objective functions at the upper and lower level are represented by F and f respectively. Inequality constraints at the upper and lower levels are represented by G_j and g_i respectively. For brevity, equality constraints have not been shown in the formulation. X_U and X_L are respectively the bound constraints for the upper level decision vector and lower level decision vector.

Bilevel optimization problems arise in several application fields such as Knapsack problems, planning problems, vehicule routing problems, scheduling problems, etc. In fact, [Dempe and Richter, 2000] proposed a bilevel forumulation for the Knapsack Problem (KP) where the leader, or the upper-level authority fixes the knapsacks capacity in order to maximize his own profit and the follower, or the lower-level authority faces a 0-1 knapsack problem implicating the capacity set by the leader. [Lukač et al., 2008] modeled the production planning problem with sequence dependent setups problem as a bilevel mixed 01 integer programming problem, where the leader was represented by the senior manager and the follower was represented by the middle manager whose main role is to control the machines. The leader of the bilevel problem minimizes the sum of the total setup time assigning the products to the machines. After the leaders decision about the setup time reduction, the follower tries to minimize his production, storage and setup cost of the machines. According to the literature, very few researches were made for scheduling problems. [Kis and Kovács, 2012] proposed a bilevel formulation for the parallel machine problem with additional constraints, where the upper-level assigns the jobs to parallel machines and the follower arranges the assigned jobs. The leader and the follower minimize together the total weighted completion time. [Ben Younes et al., 2019] propose a hybrid bilevel optimization algorithm based on both exact and approximate methods to solve the FJSP where the upper level optimizes the assignment problem using a genetic algorithm and the lower level optimizes the sequencing problem via exact method.

To our knowledge, no research exists till date where the FJSP was solved using bilevel mathematical model. The motivation behind the use of such techniques for solving the FJSP is that bilevel models help to accelerate the diversity and convergence of the search space through promising areas. In fact, more job assignments are explored in the upper-level. Then, one fixed assignment is taken into account in order to find the optimal scheduling. Moreover, the use of this new and efficient technique with an exact approach makes the search easy and helps to get optimal solutions.

The remainder of this paper is organized as follows: At first, we will begin by defining our considered FJSP in Section 2. Then, section 3 is dedicated to the presentation of our novel exact method that uses two levels of decision and relies on a mathematical model. Finally, we present the results obtained by the experimental study conducted for the proposed bilevel model and possible future prospects.

2 Problem definition

The Flexible Jobshop Scheduling Problem (FJSP) is a generalization of the classical Jobshop Scheduling Problem (JSP). The problem can be stated as follows. There is a set of n jobs and a set of m machines. Each job i is composed of n_i operations. Each operation O_{ij} is assigned to an available machine from a set of alternative machines M_{ij}, and requires a processing time P_{ij}. All the jobs are available at time zero. Operation interruption is not allowed, and each machine can perform at most one operation at the same time. The aim of this work is to find a schedule which minimizes the total processing time of all the jobs makespan or C_{max}. According to [Kacem et al., 2002], the FJSP presents two types of flexibility: i. Total Flexibility (T-FJSP), each operation

can be processed on any machine of M. In this case, M_{ij} is equal to M (M_{ij} = M). ii. Partial flexibility, each operation can be processed on one machine of subset of M machines. In this case, M_{ij} is included in M ($M_{ij} \subset M$).
The FJSP covers two Sub-problems: (i)operation-machine assignment problem which consists in assigning operations to a machine and (ii)operation sequencing problem, is to determine the processing order of operations on machines. The FJSP solved as bilevel optimization problem differs from the classical FJSP by the fact that it includes two levels of optimization task. The first level is used for assignment process, and the second level is responsible for scheduling process. Each level tries to optimize its own objective function and is influenced by the actions of the other level.
The following assumptions are considered:

- There is a set of n jobs to be performed on a set of m machines noted as$M = M_1, M_2, ..., M_m$.
- Each job is composed of a sequence of ni operations as: $(O_{i1}, O_{i2}, .., O_{ini})$.
- Each operation j of a job i (O_{ij}) requires one machine out of a set of given machines M_{ij} .
- All machines are available at time t = 0.
- All jobs are available at time t = 0.
- Each operation can be performed by only one machine or resource at any time.
- There are precedence constraints among the operations of the same job.
- Jobs are independent from each other.
- Machines are independent from each other.
- Operation preemption is not allowed (once an operation started, it cannot be interrupted).

3 Proposed bilevel model

Bilevel optimization involves two levels of optimization task where each level should satisfy some constraints and has to optimize its own objective function. The upper level is the leader, and the lower level is the follower. In this section, we will present a new bilevel mathematical formulation for the FJSP that relies on two coupled optimization problems with two decision makers, the upper level (or the leader) and the lower level (or the follower). Our considered problem is the classical FJSP with partial flexibility.

3.1 Notation

We will present in this subsection the notation used for our proposed bilevel model. The variables and the decision variables used in our formulation are defined as follows:

n	the number of jobs,
m	the number of machines,
i,i'	index for jobs where i=$\{1,2,...,n\}$ and i'=$\{1,2,...,n\}$
O_i	the set including operations of job i where $\mid O_i \mid = o_i$
j,j'	index for operations of job i where j=$\{1,...,o_i\}$ and j'=$\{1,...,o_i\}$
k	index for machines where k=$\{1, ..., m\}$,
J	the set of jobs,
M	the set of machines,
$O_{i,j}$	the j [th] operation of job i,
$e_{i,j,k}$	takes value 1 if machine i is eligible for $O_{i,j}$, and 0 otherwise,
$p_{i,j,k}$	processing time of operation $O_{i,j}$ on machine k,
M_{ij}	the set of alternative machines by which operation O_{ij} can be processed, ($M_{ij} \subseteq$ M)
$M_{ij} \cap M_{i'j'}$	the set of machines by which operations O_{ij} and $O_{i'j'}$ can be processed,
$C_{i,j,k}$	the completion time of operation $O_{i,j}$ on machine k,
$C_{i',j'}$	the completion time of operation $O_{i',j'}$,
C_i	the completion time of job i,
C_{max}	maximum completion time for all jobs (makespan),
L	a large positive number.

Decision variables:

$$\mathrm{X}_{i,j,k} = \begin{cases} 1 \text{ if machine k is selected for operation } O_{i,j}, \\ 0 \text{ otherwise.} \end{cases} \quad (2)$$

$$\mathrm{Y}_{i,j,i',j',k} = \begin{cases} 1 \text{ if operation } O_{i,j} \text{ precedes operation } O_{i',j'} \text{ on machine k}, \\ 0 \text{ otherwise.} \end{cases} \quad (3)$$

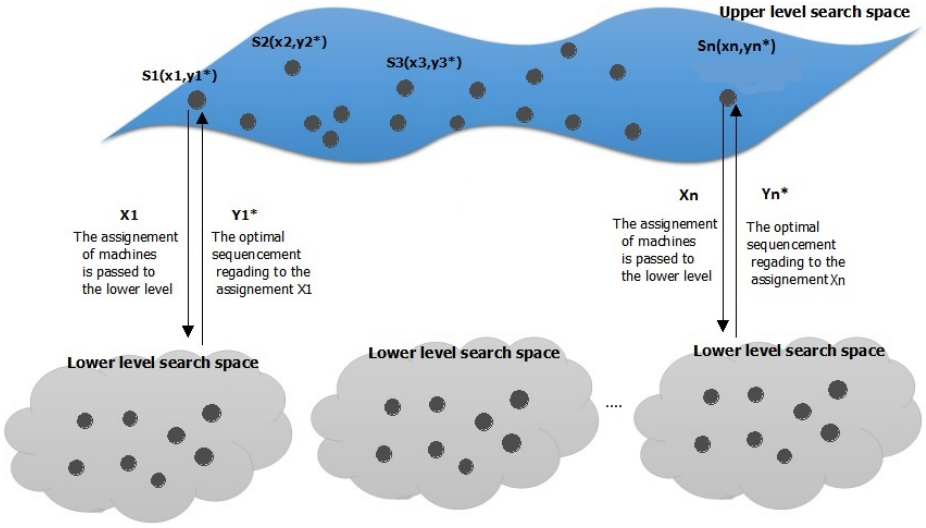

Fig. 1. FJSP modelled as bilevel optimization problem.

3.2 Bilevel Mathematical Model

The FJSP may be solved by firstly assigning the operations to a potential resource, and secondly sequencing them on the available resources. Following the above description, we introduce in this section a hierarchical structure of the classical FJSP problem with consideration of the presented assumptions and constraints. The bilevel modeling of our considered FJSP contains two levels of decision making where the upper level's (leader) objective is to assign operations to one of the alternative machines, and the lower-level's (follower) objective is to find the best sequencing of operations in a way that it minimizes the leader's objective function. In other words, the leader assigns operations to machines and the follower sequences these operations that are already assigned. Furthermore, the leaders objective depends on its own variables. But, a solution to a bilevel problem depends on both leader's and followers decisions. In other words, in order to obtain a feasible solution, the leader should take into account the follower's response. In fact, the first player (the leader) makes its selection first and communicates it to the second player (the follower). Then, knowing the choice of the leader, the follower selects its response as an optimal solution of the follower's decision problem and gives this back to the leader. Thus, the leaders task is to determine a best decision, that is, a point x* which is feasible for the leader's problem, minimizing, together with the response y*, the leader's objective function denoted as F(x,y) as shown in Figure 1.
We should remind that the major motivation behind the use of the bilevel modeling of our considered FJSP is the acceleration of the diversity and convergence of the search space by discovering all the possibilities of job assignments in the upper level in first place and focusing on one assignment from the list of all explored assignments in second place to get the best operation sequencing (optimal sequencing) that corresponds to the already fixed assignment. Therefore, the basis of the proposed scheme is to stim-

ulate the search space capacity on both levels of the FJSP decision space to obtain better solutions. We should mention that we consider the same objective function for the leader ad the follower which is the makespan. The FJSP mathematical formulation of our proposed bilevel model for the FJSP Problem is stated as follows:

Minimize$_{x_{i,j,k},y_{i,j,i',j',k}}C_{max}$

St

$$\sum_{k\in M_{ij}} X_{i,j,k} = 1, \qquad \forall i \in J, \forall j \in O_i \tag{1}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{o_i} X_{i,j,k} \leq e_{i,j,k}, \qquad \forall k \in M_{ij}, \tag{2}$$

where, for each given $\{\mathrm{x}_{i,j,k}\}, \{y_{i,j,i',j',k}\} solves$:

Minimize$_{y_{i,j,i',j',k}}$C$_{max}$,

s.t

$$\sum_{i=1}^{n}\sum_{j=1}^{o_i} Y_{i,j,i',j',k} \leq 1\ , \qquad \forall i' \in J, \forall j' \in O_{i'}, \forall k \in M_{ij} \cap M_{i'j'} \tag{3}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{o_i} Y_{i,j,i',j',k} \leq \sum_{i=1}^{n}\sum_{j=1}^{o_i} Y_{i',j',i,j,k}\ , \qquad \forall i' \in J, \forall j' \in O_{i'}, \forall k \in M_{ij} \cap M_{i'j'} \tag{4}$$

$$C_{i,j,k} \geq C_{i',j'} + Y_{i,j,i',j',k} \times p_{i,j,k} - \mathrm{L} \times (1 - \sum_{k=1}^{m} Y_{i',j',i,j,k}), \qquad \forall i, i' \in J, \forall j, j' \in O_{i'}, \forall k \in M_{ij} \cap M_{i'j'} \tag{5}$$

$$C_{i,j,k} \geq \sum_{k\in M_{i,j-1}} C_{i,j-1,k} + \sum_{i=1}^{n}\sum_{j=1}^{o_i}\sum_{k=1}^{m} Y_{i,j,i',j',k} \times p_{i,j,k} - L \times p_{i,j,k}, \qquad \forall i \in J, \forall j \in O_i \tag{6}$$

$$C_i \geq \sum_{k\in M_{ij}} C_{i,j,k}, \qquad \forall i \in J, \forall j \in O_i, \forall k \in M_{ij} \cap M_{i'j'} \tag{7}$$

$$C_{max} \geq C_i, \qquad \forall i \in J \tag{8}$$

$$C_{i,j,k} \geq 0, \qquad \forall i \in J, \forall j \in O_i, \forall k \in M_{ij} \tag{9}$$

The proposed bilevel model has two parts, or levels, that are used for the assignment process and the scheduling process, respectively. Thus, the description of our bilevel mathematical formulation is stated as follows:
Constraint (1) ensures that each operation should be assigned to one machine. Moreover, constraint (2) makes sure that each operation is assigned to one of its eligible machines. Constraint (3) is used to verify that each operation could have at most one succeeding operation. Constraint (4) ensures that each operation should have exactly one preceding operation on the same machine. Constraint (5) is used to make sure that one machine cannot execute two different operations simultaneously and an operation cannot be executed on two distinct machines at the same time. In fact, the difference between the completion times of two succeeding operations should be greater than the processing time of the operation performed earlier. Constraint (6) guarantees the precedence constraints between two operations $O_{i,j}$ and $O_{i,j'}$ of job i. So that, operation $O_{i,j}$ can start its processing only if operation $O_{i,j-1}$ is completed. Constraint (7) represents the completion time of job i. Constraint (8) is used to verify that the total completion time for all jobs is greater than or equal to the completion time for each job i. The last constraint (9) forces the completion time of operation $O_{i,j}$ processed on machine k to be positive.
Constraints (1) and (2) represent the upper-level. The other constraints (constraints (3), (4), (5), (6), (7), (8), (9), and (10)) represent the lower-level. In fact, the leader searches for the best operation assignment and communicates it to the follower. According to the leader's choice, the follower makes its decision and returns an optimal solution which is an optimal scheduling of the fixed assignment as a response to the leader's selection. Both leader and follower attempt to minimize the total completion time objective function (C_{max}).

4 Experimentation and Results

In this section,we analyze the impact of using bilevel structure to solve the FJSP. We perform experiments with CPLEX Optimization Studio v12.7.1. All experiments were performed on an Intel Quad Core 2.16 GHz machine (in 64 bit mode) with 4GB memory. We use commonly well known benchmarks proposed in the literature for the FJSP are presented: BRdata: proposed by [Brandimarte, 1993], and HUdata: proposed by [Hurink et al., 1994]. These different benchmarks are extracted from the library FJS-PLIB whose address is: www.idsia.ch/ monaldo/fjspresults/textdata.zip.

We compare our proposed model with a monolevel mathematical formulation with the same used constraints. So, we implement a monolevel model with the same constraints as in our bilevel model. We also compared the results obtained by our model with other exact and approximate approaches proposed in the literature as follow: The published resultsof [Mastrolilli and Gambardella, 2000] which are in the form of an interval [LB; UB]. LB denotes the lower bound and UB denotes the upper bound. In the case where LB=UB then the result corresponds to the optimal solution. An exact method proposed by [Behnke and Geiger, 2012], which is based on constraint programming; a hybridization between GA and TS called MAS-GATS model based on Multi

Name	n	m	LB	UB	FJS MAT-SLO+	MAS-GATS	Exact method [Behnke et al., 2012]	MILP Model (Mono-level)	MILP Model (Bi-level)
MK01	10	6	36	42	40	39	40	38	36*
MK02	10	6	24	32	32	28	27	27	24*
MK03	15	8	204	211	207	204	204	204	204*
MK04	15	8	48	81	67	70	60	60	60*
MK05	15	4	168	186	188	176	174	174	171*
MK06	10	15	33	86	59	85	75	59	59*
MK07	20	5	133	157	154	148	143	143	141*
MK08	20	10	523	523	523	529	523	523	523*
MK09	20	10	299	369	437	349	307	307	307*
MK10	20	15	165	296	380	278	214	214	209*

Table 1. The results of C_{max} obtained on BRdata benchmarks [Brandimarte., 1993].

Agent model introduced by [Azzouz et al., 2015]; . An hierarchical model denoted as FJS MATSLO+, proposed by [Ennigrou and Ghedira, 2008], which combines Multi-Agent System with Tabu search (TS) using diversification techniques, as well as two terminals LB and UB existing in the literature. More recently, a bilevel hybrid evolutionary method proposed by [Ben Younes et al., 2019].

Tables 1 and 2 show the results obtained by our model compared with the monolevel model and existing approaches from the literature in term of makespan objective function. The star brand (*) means that the result obtained by our proposed model is better than the basic model. The tables also show the size of the benchmarks used in terms of the number of jobs n, the number of resources m.

From these tables, we remark that the objective function C_{max} for all test Brandimart and Hurink instances of the Bi-level MILP Model, is better than the results obtained by the Mono-level MILP Model, as well as the exact and approximate approaches. From these results, we state that the bilevel structure keeps its robust performance for all size of bechmarks.

Moreover, in order to further evaluate the performance of our Bi-level scheme for the proposed algorithm, we compared the algorithms in term of $CPUtime$. Table 3 and

Name	n	m	HGA[Ben Younes et al., 2019]	LB	UB	FJS MAT-SLO+	MAS-GATS	Exact method [Behnke et al., 2012]	MILP Model (Mono-level)	MILP Model (Bi-level)
la01	10	5	609	609	609	609	621	609	609	609*
la02	10	5	655	655	655	655	704	655	655	655*
la03	10	5	550	550	554	575	578	567	554	550*
la04	10	5	568	568	568	579	607	568	568	568*
la05	10	5	503	503	503	503	524	503	503	503*
la06	15	5	833	833	833	858	833	833	833	833*
la07	15	5	762	762	765	778	813	765	765	762*
la08	15	5	845	845	845	845	860	845	845	845*
la09	15	5	878	878	878	890	891	878	878	878*
la10	15	5	866	866	866	871	873	866	866	866*
la11	10	20	1103	1078	1103	1135	1129	1106	1103	1078*
la12	20	5	960	960	960	982	960	960	960	960*
la13	20	5	1053	1053	1053	1067	1053	1053	1053	1053*
la14	20	5	1123	1123	1123	1144	1137	1123	1123	1123*
la15	20	5	1111	1111	1111	1194	1229	1111	1111	1111*
la16	10	10	892	892	915	896	1019	915	915	892*
la17	10	10	707	707	707	707	773	707	707	707*
la18	10	10	842	842	843	845	914	843	843	842*
la19	10	10	799	796	796	813	937	799	796	796*
la20	10	10	863	857	864	863	886	857	857	857*

Table 2. The results of C_{max} obtained on the HUdata Benchmarks of [Hurink et al., 1994] set edata.

Instance problem	n	m	Mono-level (CPU)	Bi-level (CPU)
MK01	10	6	102,8	42.6
MK02	10	6	120	48.2
MK03	15	8	189.6	166.5*
MK04	15	8	135.3	78*
MK05	15	4	125.6	60*
MK06	10	15	354.7	206.3*
MK07	20	5	301.3	156*
MK08	20	10	496	253*
MK09	20	10	556	336.6*
MK10	20	15	906.6	632*

Table 3. The results of $CPUtime$ obtained on the Brandimart BRdata benchmarks.

Table 4 show the obtained results by the Bi-level MILP model and the Mono-level MILP model in term of $CPUtime$. Follwing these tables, we prove that the Bi-level Model gives better results than the Mono-level model in terms of $CPUtime$. In fact, The Bi-level MILP model is faster as it is able to be executed in less than 632 s for all the given instances, while the time required by the mono-level model to execute each test instance reaches at most 906,6 s. Following Table 4, we remark that our bilevel model outperformed the hybrid bilevel model proposed by [Ben Younes et al., 2019] especially in the big size instances in term of CPU Time.
Furthermore, we study the performance of the proposed algorithm regarding to the problem size. Figure 2 shows the obtained results in term of C_{max} for the two kinds of benchmarks respectively. Figure 3 presents the obtained performance against the number of machines. Acoording to the figures, we remark that our algorithm keeps its robust performance in different problem sizes. Figures 4-5 show the average of CPU according to the number of jobs and machines for both benchmarks. It proves that our bilevel proposed model obtains better CPU than the classical model for all test instances.
The Bi-level MILP Model is able to find better solutions as compared to the Mono-level MILP Model. Furthermore, the Mono-level MILP Model requires more computational time to run the test instances. In fact, our bilevel proposed model is better than the basic model in terms of CPU as it requires less computational time than the basic model for all test instances.

Instance problem	n	m	CPU [Ben Younes et al.,2019]	CPU (Mono-level)	CPU (Bi-level)
La01	10	5	9.97*	62.5	28.5
La02	10	5	18	53.4	17.5*
La03	10	5	35	32.4	21.8*
La04	10	5	32	35	10.7*
La05	10	5	9*	40	10.4
La06	15	5	15	82	14.6*
La07	15	5	29*	95.7	33
La08	15	5	19	92	18*
La09	15	5	20*	78.6	31.8
La10	15	5	12*	68.6	16.8
La11	20	5	26*	102.5	37
La12	20	5	21	90.6	18.8*
La13	20	5	26	89	12*
La14	20	5	25*	98.6	35.2
La15	20	5	69	96.4	15*
La16	10	10	57	66.6	20.2*
La17	10	10	54	52.6	15.6*
La18	10	10	60	87.3	26*
La19	10	10	61	53.8	13.7*
La20	10	10	60	59.3	13.7*

Table 4. The results of $CPUTime$ obtained on the HUdata Benchmarks of [Hurink et al., 1994] set edata.

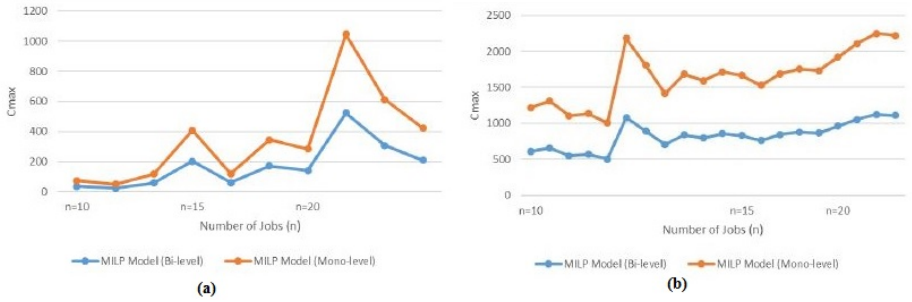

Fig. 2. Comparison of C_{max} VS. Number of Jobs (n) for two benchmarks: (a) [Brandimarte, 1993] and (b)[Hurink., 1994] set edata.

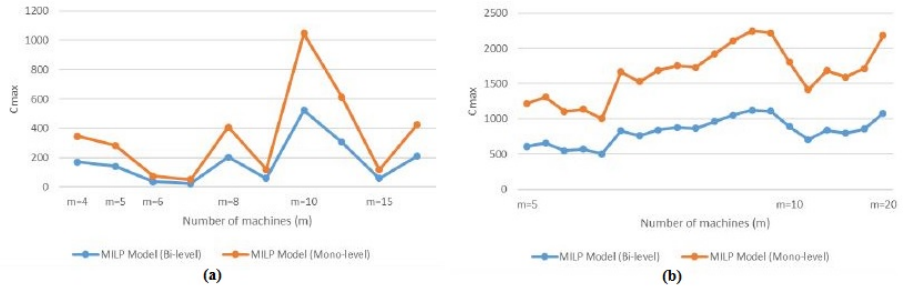

Fig. 3. Comparison of C_{max} VS. Number of machines (m) for two benchmarks: (a) [Brandimarte, 1993] and (b)[Hurink., 1994] set edata.

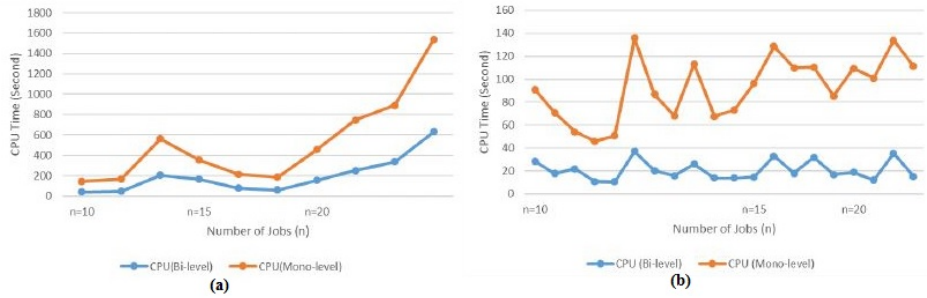

Fig. 4. Comparison of $CPUtime$ VS. Number of Jobs (n) for two benchmarks (a)[Brandimarte, 1993] and (b) [Hurink., 1994] set edata.

5 Conclusion and future research

We developed an effective and efficient bilevel mathematical model where each level is executed separately. In fact, each operation is assigned to one of its eligible machines first. Then, for each given assignment, the corresponding scheduling is performed. Gen-

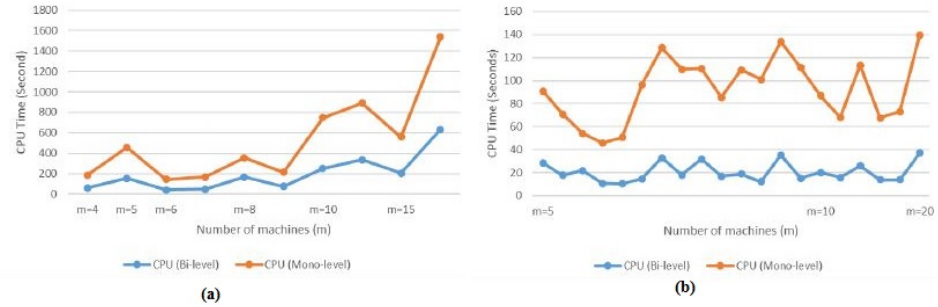

Fig. 5. Comparison of $CPUtime$ VS. Number of machines (m) for two benchmarks: (a)[Brandimarte, 1993] (b) [Hurink., 1994] set edata.

erally, the assignment and the scheduling of the different operations are made simultaneously. The interest of using such bilevel framework is that it facilitates the search for solutions and enhances the search space to explore more promising areas in both levels. In fact, each level cooperates with the other to provide solutions for the overall problem in order to identify optimum solutions with low computational cost. For future directions, we will use bilevel framework in order to solve more complex scheduling problems.

References

[Azzouz et al., 2015] Azzouz, A., Ennigrou, M., and Jlifi, B. (2015). Diversifying ts using ga in multi-agent system for solving flexible job shop problem. In *Proceedings of the 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO) vol1,pp 94-101*.

[Behnke and Geiger, 2012] Behnke, D. and Geiger, M. J. (2012). Test instances for the flexible job shop scheduling problem with work centers.

[Ben Younes et al., 2019] Ben Younes, H., Azzouz, A., and Ennigrou, M. (2019). Solving flexible job shop scheduling problem using hybrid bilevel optimization model. In *In: Madureira A., Abraham A., Gandhi N., Varela M. (eds) Hybrid Intelligent Systems. HIS 2018. Advances in Intelligent Systems and Computing, vol 923. Springer, Cham*.

[Bracken and McGill, 1973] Bracken, J. and McGill, J. T. (1973). Mathematical programs with optimization problems in the constraints. *Operations Research*, 21(1):37–44.

[Brandimarte, 1993] Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations research*, 41(3):157–183.

[Candler and Norton, 1977] Candler, W. and Norton, R. (1977). *Multilevel programming. Technical Report 20, World Bank Development Research*. Washington D. C.

[Dempe and Richter, 2000] Dempe, S. and Richter, K. (2000). *Bilevel programming with knapsack constraints*. TU Bergakademie, Fakultät für Mathematik und Informatik.

[Ennigrou and Ghedira, 2008] Ennigrou, M. and Ghedira, K. (2008). New local diversification techniques for flexible job shop scheduling problem with a multi-agent approach. In *In Autonomous Agents and Multi-Agent Systems, vol.17(2),270-287*.

[Garey et al., 1976] Garey, M. R., Johnson, D. S., and Stockmeyer, L. (1976). Some simplified np-complete graph problems. In *Theoretical computer science, 1(3), 237-267*.

[Hurink et al., 1994] Hurink, J., Jurisch, B., and Thole, M. (1994). Tabu search for the job shop scheduling problem with multi-purpose machines. In *Operations-Research-Spektrum,15(4):205-215*.

[Kacem et al., 2002] Kacem, I., Hammadi, S., and Borne, P. (2002). Approach by localization and multi-objective evolutionary optimization for flexible job-shop scheduling problems. In *Syst IEEE Syst Man Cybern 32(1):113*.

[Kis and Kovács, 2012] Kis, T. and Kovács, A. (2012). On bilevel machine scheduling problems. *OR spectrum*, 34(1):43–68.

[Lukač et al., 2008] Lukač, Z., Šorić, K., and Rosenzweig, V. V. (2008). Production planning problem with sequence dependent setups as a bilevel programming problem. *European Journal of Operational Research*, 187(3):1504–1512.

[Mastrolilli and Gambardella, 2000] Mastrolilli, M. and Gambardella, L. M. (2000). Effective neighbourhood functions for the flexible job shop problem. *Journal of scheduling*, 3(1):3–20.

[Mousakhani, 2013] Mousakhani, M. (2013). Sequence-dependent setup time flexible job shop scheduling problem to minimise total tardiness. In *International Journal of Production Research, 51(12), 3476-3487*.

[Özgüven et al., 2010] Özgüven, C., Özbakır, L., and Yavuz, Y. (2010). Mathematical models for job-shop scheduling problems with routing and process plan flexibility. *Applied Mathematical Modelling*, 34(6):1539–1548.

[Saidi-Mehrabad and Fattahi, 2007] Saidi-Mehrabad, M. and Fattahi, P. (2007). Flexible job shop scheduling with tabu search algorithms. In *The International Journal of Advanced Manufacturing Technology, 32(5-6), 563-570*.

[von Stackelberg, 1934] von Stackelberg, H. (1934). Marktform and gleichgewicht, julius springer, vienna, austria,.