Computers & Industrial Engineering 142 (2020) 106347

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem

Leilei Meng[a,b], Chaoyong Zhang[b,*], Yaping Ren[c], Biao Zhang[a], Chang Lv[b]

[a] School of Computer Science, Liaocheng University, Liaocheng 252059, China
[b] State Key Lab of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan 430074,China
[c] School of Intelligent Systems Science and Engineering, Jinan University (Zhuhai Campus), Zhuhai 519070, China

ARTICLE INFO

Keywords:
Distributed flexible job shop scheduling problem
Mixed integer linear programming
Constraint programming
Makespan

ABSTRACT

This paper intends to address the distributed flexible job shop scheduling problem (DFJSP) with minimizing maximum completion time (makespan). In order to solve this problem, we propose four mixed integer linear programming (MILP) models as well as a constraint programming (CP) model, among which four MILP models are formulated based on four different modeling ideas. MILP models are effective in solving small-scaled problems to optimality. DFJSP is NP-hard, therefore, we propose an efficient constraint programming (CP) model based on interval decision variables and domain filtering algorithms. Numerical experiments are conducted to evaluate the performance of the proposed MILP models and CP model. The results show that the sequence-based MILP model is the most efficient one, and the proposed CP model is effective in finding good quality solutions for the both the small-sized and large-sized instances. The CP model incomparably outperforms the state-of-the-art algorithms and obtains new best solutions for 11 benchmark problems. Moreover, the best MILP model and CP model have proved the optimality of 62 best-known solutions.

1. Introduction

In recent years, with increasingly decentral and global economy, a multi-factory environment becomes more and more important. Therefore, the production scheduling in multi-factory production environment, referred to as the distributed scheduling problem, has received more and more concerns (Li et al., 2019; Pan, Gao, Wang, Liang, & Li, 2019; Ruiz, Pan, & Naderi, 2019). Among all types of distributed scheduling problems, the distributed flexible job shop scheduling problem (DFJSP) represents a multi-factory environment, in which each factory is a flexible job shop (Nouiri, Bekrar, Jemai, Niar, & Ammari, 2018; Sun, Lin, Li, & Gen, 2019; Wu, Liu, & Zhao, 2018). Each job must be allocated to exactly one factory (De Giovanni & Pezzella, 2010). Obviously, in DFJSP, three sub-problems must be solved: (1) determining the most suitable factory for each job (factory selection problem), (2) determining the machine selection of all operations in each factory (machine selection problem) and (3) determining the operation sequence on each machine in all factories (sequencing problem). Obviously, DFJSP is a more difficult NP-hard problem than the traditional FJSP with only one factory.

When studying a scheduling problem especially a new one, it is very important to formulate a feasible mathematical model for it. This is because mathematical models could solve the relatively small-sized problems to optimality and further help mine the inherent information of the scheduling problem. Moreover, the optimal solutions can be used as the reference standard to develop approximate methods (Li, Sang, Han, Wang, & Gao, 2018). Therefore, in this paper, we formulate four MILP models for solving DFJSP based on four modeling ideas, namely, sequence-based idea, position- based idea, time-indexed idea and adjacent sequence-based idea.

Due to inherent NP hardness of DFJSP, the MILP model is not effective to solve the relatively large-sized problems. Since constraint programming (CP) shows good performances on scheduling problem with makespan objective, we develop a CP formulation to solve DFJSP effectively. CP performs well at exploring feasible solutions in short computational times and hence allow to find good quality solutions. This is because CP is adapt at expressing complex relationships in terms of global constraints. Moreover, effective domain filtering algorithms are used for solving these constraints in CP. Compared to other exact methods such as mathematical programming (MP), CP requires much less number of decision variables and constraints. The effectiveness of the CP method has been proven over many combinatorial problems such as parallel machine scheduling (Gedik, Kalathia, Egilmez, & Kirac, 2018; Gedik, Rainwater, Nachtmann, & Pohl, 2016), production

* Corresponding author.
E-mail address: zcyhust@hust.edu.cn (C. Zhang).

https://doi.org/10.1016/j.cie.2020.106347
Received 5 September 2019; Received in revised form 22 December 2019; Accepted 7 February 2020
Available online 11 February 2020
0360-8352/ © 2020 Published by Elsevier Ltd.

scheduling (Novara, Novas, & Henning, 2016), and assembly line balancing problems (Bukchin & Raviv, 2018). Moreover, it has been proved that CP method is effective for solving both small-sized and large-sized problems (Gedik et al., 2016, 2018). Compared with the existing research, the contributions of this work can be concluded as follows:

(1) To the best of our knowledge, this work is the first attempt to solve DFJSP with exact algorithms namely MILP model and CP model.
(2) Four different MILP models based on four different modeling ideas are proposed and evaluated. Due to inherent NP hardness of DFJSP, the MILP models is not effective to solve the relatively large-sized problems. Thus, a CP model is proposed to solve DFJSP.
(3) The MILP models and CP model are compared with state-of-the-art algorithms for solving DFJSP. The experimental results show that the MILP models and CP model explore the optimal solutions for 62 benchmark problem instances, for which the optimal solutions were previously unknown/unproven. The determination of optimal solution is very meaningful for designing approximate methods.
(4) The CP model is very effective and outperforms all the state-of-the-art algorithms. Moreover, due to the simplicity of the CP formulation and the ease of using the currently available solver IBM CP Optimizer, CP method is very suitable for practitioners to implement in practice.

The rest of this paper is organized as follows: Section 2 presents the literature review of DFJSP. Then, in Section 3, we formulate four MILP models based on different modeling ideas. Section 4 introduces the CP formulation. Section 5 shows the experimental results. Finally, conclusion and future study are given in Section 6.

2. Literature review

The literature review of this paper is divided into three parts, among which the first one reviews the related researches about DFJSP, the second one presents the related studies about MILP modeling for scheduling problems and the last one shows the related studies about constraint programming approach for scheduling problems.

2.1. Literature review about DFJSP

In order to minimize makespan of distributed job shop scheduling problem (DJSP), Jia, Nee, Fuh, and Zhang (2003) proposed a modified genetic algorithm (MGA), in which the encoding of chromosome was designed straightforward based on classic job-shop problems without considering the machine flexibility. For dealing with DJSP, Naderi and Azab (2014) proposed two different MILP models on the basis of sequence-based and position-based modeling ideas at the first time. The two MILP models could solve small-sized problems to optimality with using CPLEX, and the results showed the sequence-based MILP model was effective for solving DJSP (Naderi & Azab, 2014).

As far as we know, Chan, Chung, and Chan (2005) firstly investigated the DFJSP. A genetic algorithm with dominant genes (GADG) was proposed to solve DFJSP. In addition, a new chromosome encoding (called SCH) was proposed, where genes specify all the scheduling information namely job, operation, machine and factory. Thus, the encoding can determine all the three sub-problems of DFJSP. Besides, in GADG, one crossover operator, two mutation operators, elitist strategy, and similarity checking were designed contrapuntally. Then, Chan, Chung, Chan, Finke, and Tiwari (2006), Chan, Chung, and Chan (2006) extended the proposed GADG to solve the DFJSP with considering machine maintenance. Moreover, based on GADG, Chung, Chan, and Chan (2009) proposed a modified GADG with an improved local search method to decide the job sequence and machine maintenance.

For optimizing makespan of DFJSP, De Giovanni and Pezzella (2010) developed an improved genetic algorithm (IGA), in which the chromosome representation, referred to as SGP, models only two DFJSP decisions (namely assignment problem and sequencing problem) and the routing problem is determined by a heuristic rule. Moreover, a new local search method based on critical factory was proposed, and it was further implemented in references (Lu, Wu, Tan, Peng, & Chen, 2015; Wu, Lin, Lin, & Chen, 2017). Lu et al. (2015) proposed a GA embedded with a concise chromosome representation (GA_JS), in which the chromosome representation, referred to as SJS, presents DFJSP solutions only by job sequences. The three sub-problems were determined by three heuristic rules in the decoding process. Chang and Liu (2017) developed a hybrid genetic algorithm (HGA), in which the chromosome representation, referred to as SCL, can explicitly decide all the three sub-problems. In SCL encoding, each gene is a float number, the integer part and decimal part of which represent the operation sequence and the machine-factory respectively. Wu et al. (2017) analysed the existing chromosome representations and decoding methods, and proposed a GA with a chromosome representation (called SOP), in which only the sequencing problem is decided. The other two DJFSP sub-problems were determined by two heuristic rules in the decoding respectively. More recently, Lin, Lee, and Wu (2019) proposed two incomplete chromosome representations (SG1 and SG2) and four GAs for solving DFJSP with considering machine maintenance. Moreover, an effective methodology for generating new chromosomes (called shadow chromosomes) from good quality solutions was proposed to improve the performance of the proposed GAs. The experimental results showed that the GA with SG2 encoding scheme and the method for generating shadow chromosomes outperformed the other three GAs and two existing state-of-the-art algorithms.

Ziaee (2014) proposed a fast-constructive heuristic algorithm for solving DFJSP. The computational results showed that the heuristic was more computationally efficient than IGA, while the latter outperformed the former in terms of best solution. Wu, Guo, Li, and Wang (2018) proposed an improved differential evolution simulated annealing algorithm to solve DFJSP with minimizing the earliness/tardiness and the total cost simultaneously. Other meta-heuristic algorithms such as chemical reaction optimization (CRO) algorithm (Marzouki, Driss, & Ghédira, 2018), artificial bee colony algorithm (ABC) (Wu et al., 2018), and particle swarm optimization (PSO) algorithm (Huang & Yao, 2012) have been also tried to solve DFJSP.

Obviously, existing research about DFJSP mainly focuses on meta-heuristic and heuristic algorithms, especially GA. They are approximate methods and cannot guarantee the optimal solutions even for small-sized problems. Moreover, the performances of meta-heuristic algorithms depend heavily on encoding and decoding rules. For a meta-heuristic algorithm, different encoding and decoding rules will show great difference on performance. If they are not well designed, the meta-heuristic algorithms may perform very poorly. Therefore, exact algorithms and other easily implemented algorithms are worthy of research. In this paper, we propose to use two exact methods namely MILP method and CP method to solve DFJSP, and explore their advantages and disadvantages for dealing with DFJSP with minimizing makespan.

2.2. Literature review about MILP modeling of scheduling problems

In recent years, with the rapid development of computer performance as well as the advent of several efficient optimization softwares (e.g., CPLEX and GUROBI), an increasing number of research has been paying attention to MILP modeling for scheduling problems (Ren et al., 2020, 2017). The MILP models used for scheduling problems can be classified into four major categories. These categories are position-based modeling idea (Wagner, 1959), sequence-based modeling idea (Manne, 1960), time-indexed modeling idea (Brandimarte, 1993) and adjacent sequence-based modeling idea (Mousakhani, 2013). The main difference between these models lies in the definition of the variables used to determine the sequencing problem. Position-based modeling method divides a machine into several processing positions according to the time sequence, and introduces variables for deciding whether an operation is assigned to a position of a machine. With regard to sequence-based modeling idea, it defines decision variables that determine the precedence relationship between two different operations (adjacent or non-adjacent) on the same machine. Time-indexed modeling idea divides the total processing time into processing periods according to a certain time interval, and binary decision variables that determine whether a certain time period on one machine is used to process some operation are introduced. The adjacent sequence-based modeling idea introduces the decision variables that determine the precedence relationship between two adjacent operations on the same machine.

Based on sequence-based modeling idea, a MILP model was firstly proposed by Özgüven, Özbakır, and Yavuz (2010) for FJSP, and then improved MILP models were developed in reference (Roshanaei, Azab, & ElMaraghy, 2013). Demir and Kürşat İşleyen (2013) compared the MILP model proposed by Özgüven et al. (2010) with four non-linear models. The experimental results showed that the linear model outperformed the four non-linear models. Also based on sequence-based idea, Shen, Dauzère-Pérès, and Neufeld (2018) developed a MILP model for solving FJSP with sequence-dependent setup times (SDST), and Naderi, Ghomi, Aminnayeri, and Zandieh (2011) proposed a MILP model for solving flexible open shop scheduling problem.

On the basis of position-based modeling idea, Fattahi, Saidi Mehrabad, and Jolai (2007) established a MILP model for FJSP with minimizing makespan for the first time, in which two binary decision variables were used. Then, an improved MILP model with only one binary decision variable and less constraints was proposed (Roshanaei et al., 2013). Zhang, Tang, Wu, and Wang (2017) proposed a MILP for FJSP with minimizing energy consumption, and Meng, Zhang, Shao, and Ren (2019) proposed six more efficient MILP models. For solving hybrid flow shop scheduling problem with unrelated parallel machines (HFSP-UPM) with minimizing energy consumption, Meng, Zhang, Shao, Ren, and Ren (2019) formulated five MILP models with different decision variables. In order to solve multi-objective HFSP-UPM with simultaneously minimizing makespan and energy consumption, Zhang et al. (2019) proposed a multi-objective MILP model based on weight coefficient method. Taking worker flexibility into consideration, Meng, Zhang, Zhang, and Ren (2019) formulated two MILP models based on different modeling ideas.

Based on time-indexed modeling method, Wu and Chien (2008) formulated a MILP model for semiconductor final test scheduling problem, which is an extension of classical FJSP with considering machine-dependent setup times and other limited resource constraints. Also based on this idea, Gicquel, Hege, Minoux, and van Canneyt (2012) proposed a MILP for HFSP with limited-wait constraints.

Based on adjacent sequence-based modeling method, Mousakhani (2013) proposed a MILP model for FJSP with minimizing total tardiness. Also based on this idea, Vallada and Ruiz (2011) developed a MILP model for unrelated parallel machine scheduling problem with sequence dependent setup times (UPM-SDST) with minimizing makespan, Naderi, Zandieh, and Shirazi (2009) proposed a MILP model for solving HFSP with total weighted tardiness minimization, Keskinturk, Yildirim, and Barut (2012) developed a MILP model for UPM-SDST with optimizing load balancing, and Ruiz, Şerifoğlu, and Urlings (2008) proposed a MILP model for solving realistic HFSP with some jobs skipping stages.

Based on position-based, sequence-based and time-indexed modeling methods, Pan (1997) proposed different mathematical models of job shop scheduling problem (JSP) and flow shop scheduling problem (FSP). Also based on these three modeling ideas, Demir and Kürşat İşleyen (2013) analyzed and compared different (linear and non-linear) mathematical models of flexible job shop (FJSP). Similarly, based on these three ideas, Naderi, Fatemi Ghomi, Aminnayeri, and Zandieh (2011) proposed four MILP models for open shop scheduling to minimize total tardiness. For dealing with DJSP, Naderi and Azab (2014) proposed two MILP models based on position-based and adjacent sequence-based modeling methods respectively. Besides, Naderi, Khalili, and Khamseh (2013) proposed two MILP models for no-wait flowshop scheduling with parallel machines on the basis of sequence-based and adjacent sequence-based ideas respectively. Based on sequence-based and position-based ideas, Rocha, Ravetti, Mateus, and Pardalos (2008) developed two MILP models to solve UPM-SDST. Based on time-indexed and adjacent sequence-based modeling methods, Matta (2009) proposed two MILP models for flexible open shop problem. Based on the four modeling ideas, Meng, Zhang, Shao, Zhang, et al. (2019) formulated several new MILP models for HFSP-UPM and its extended problems with considering sequence-dependent setup times, no-wait and blocking constraints.

Based on different modeling ideas, different decision variables and constraints will be introduced. The performance of mathematical model is mainly determined by the numbers of binary decision variables, constraints and continuous decision variables (Jin, Tang, Zhang, Shao, & Tian, 2016; Naderi & Azab, 2014; Pan, 1997). Therefore, the performances of MILP models based on different modeling ideas may differ greatly.

2.3. Literature review about constraint programming (CP) approach for scheduling problems

In recent years, with the advent of robust solvers such as IBM CP Optimizer and Gurobi, constraint programming has been widely applied to a variety of scheduling problems, which can be attributed to many reasons. Firstly, compared with MP, CP allows declarative, flexible and compact formulations, which make adding new constraints to the scheduling much more convenient and straightforward. Especially, CP has competitive advantage of representing complex constraints such as if-then and nonlinearity without reformulation over MP, which makes it perform well for finding feasible solutions with good quality. Secondly, most meta-heuristic algorithms include several optional parameters, and a fine tuning of parameters must be done to get best performance. However, CP belongs to a category of off-the-rack and does not need much fine tuning. For a practitioner, he only needs to provide a description of the studied problem (Ham & Cakici, 2016). Thirdly, CP can solve large problems effectively. It has been proved that CP is very effective in solving many discrete optimization problems.

CP method has been widely used to solve parallel machine scheduling problem. For minimizing the processing cost of unrelated parallel machine scheduling problem (UPM) with deadlines and release dates, Jain and Grossmann (2001) proposed a hybrid MILP/CP model based on decomposition method, and compared it with pure MILP model, pure CP model and combined MILP-CP model. The results showed that the hybrid MILP/CP model performed very efficiently and outperformed all the other models. Edis and Oguz (2012) proposed a hybrid MP/CP model for solving UPM with flexible resources. For solving UPM-SDST, Gedik et al. (2018) proposed an effective CP model, which explored the optimal solutions for 283 benchmark problem instances. With regard to UPM-SDST with job availability intervals, Gedik et al. (2016) proposed a combined CP/MP model based on logic-based Benders algorithms to schedule jobs on unrelated parallel machines in a fixed planning horizon. For solving the UPM with a resource constraint (UPMR), Fleszar and Hindi (2018) proposed a CP model, a MILP model for two-machine UPMR and a two-stage heuristic based on the hybrid of MILP model and CP model. Experimental results showed that the hybrid algorithm combining two-stage heuristic and a CP model outperformed the previously proposed methods and found good solutions for much larger instances.

With regard to other scheduling problems, Ku and Beck (2016) proposed a CP model for solving job shop scheduling problem (JSP) with minimizing makespan. For solving the same problem, Oliveira and

Ribeiro (2015) proposed two different CP models based on different modeling ideas and compared them with MILP model. For solving FJSP with parallel batch processing machines (FJSP-PBM), Ham and Cakici (2016) developed a CP model and compared it with MILP models. Novara et al. (2016) developed a CP model for scheduling problems in multiproduct multistage batch plants. Kelbel and Hanzálek (2011) gave an application of CP in production scheduling. Bukchin and Raviv (2018) applied the CP method to solve various assembly line balancing problems. With regard to minimizing total tardiness of multi-level job scheduling in a flexible job shop environment, Na and Park (2014) proposed a MILP model and used CP method as a compared algorithm. Öztürk, Tunalı, Hnich, and Örnek (2015) studied a problem of balancing and cyclic scheduling of flexible mixed model assembly lines with parallel stations, and proposed a novel CP model with considering problem specific symmetry breaking constraints. In order to minimize makespan of flexible job shop scheduling problem with lot streaming and lot sizing of the variable sublots, Bożek and Werner (2018) proposed a MILP model and a CP model to solve it.

3. MILP models

3.1. Notations

The notations and decision variables used in this paper are given in Table 1.

3.2. Problem definition

The DFJSP can be described as follows: there are a set of independent jobs $I = \{1, 2, \cdots, n\}$ to be processed in a set of factories $F = \{1, ..., r_f\}$. Each factory stands for a flexible job shop problem (FJSP). Each job i includes n_i operations $\{O_{i,1}, O_{i,2}, ..., O_{i,n_i}\}$ and has its own processing route. Each operation $O_{i,j}$ can be processed by a predefined subset of candidate machines in a factory $K_{i,j,f} \subseteq K_f$. Moreover, all the operations of the same job must be processed in the same factory.

Moreover, some assumptions are given as follows:

- All the factories, jobs, and machines are available at time zero.
- In every factory, a machine can process at most one operation at a time.
- For each operation, no preemption is allowed.
- All the processing data is deterministic and known in advance.
- Each operation is assigned to exactly one eligible factory and one machine of this factory.
- For modeling DFJSP more easily, the transportation time $Tr_{i,f}$ is included in the processing time of last operation $pt_{i,n_i,f,k}$.

The objective in this paper is to minimize makespan by reasonably solving the three subproblems namely factory selection problem, machine selection problem and sequencing problem. The objective is given as below, min $C_{\max}$

3.3. Model 1 (sequence- based model)

Model 1 is based on sequence-based modeling idea, and it is inspired by the MILP model for FJSP (Roshanaei et al., 2013). To solve sequencing problem, Model 1 introduces precedence decision variable $Y_{i,j,i',j'}$, which determines the precedence relationship between two operations (adjacent or non-adjacent) on the same machine (sequencing problem). $X_{i,j,f,k}$ is introduced to determine the machine selection of all operations (machine selection problem), and $Z_{i,f}$ is introduced to decide the factory selection of all jobs (factory selection problem).

3.3.1. Decision variable

Five decision variables namely $X_{i,j,f,k}$, $Y_{i,j,i',j'}$, $Z_{i,f}$, and $C_{\max}$ are included in Model 1.

3.3.2. Constraint sets

In Model 1, the following constraint sets (1)–(7) are needed.

$$\sum_{f\in F} Z_{i,f} = 1, \forall\, i \in I \tag{1}$$

$$Z_{i,f} = \sum_{k\in K_{i,j,f}} X_{i,j,f,k}, \;\; \forall\, i \in I, j \in J_i, f \in F \tag{2}$$

$$B_{i,j} + \sum_{f\in F}\sum_{k\in K_{i,j,f}} (pt_{i,j,f,k}X_{i,j,f,k}) \leqslant B_{i,j+1}, \forall\, i \in I, j \in JJ_i \tag{3}$$

$$B_{i,j} + pt_{i,j,f,k}X_{i,j,f,k} \leqslant B_{i',j'} + M\left(3 - Y_{i,j,i',j'} - X_{i,j,f,k} - X_{i',j',f,k}\right)$$
$$\forall\, i \in I, i' \in I, i < i', j \in J_i, j' \in J_{i'}, f \in F, k \in K_{i,j,f} \cap K_{i',j',f} \tag{4}$$

$$B_{i',j'} + pt_{i',j',f,k}X_{i',j',f,k} \leqslant B_{i,j} + M\left(2 + Y_{i,j,i',j'} - X_{i,j,f,k} - X_{i',j',f,k}\right)$$
$$\forall\, i \in I, i' \in I, i < i', j \in J_i, j' \in J_{i'}, f \in F, k \in K_{i,j,f} \cap K_{i',j',f} \tag{5}$$

$$C_{\max} \geqslant B_{i,n_i} + \sum_{f\in F}\sum_{k\in K_{i,n_i,f}} \left(pt_{i,n_i,f,k}X_{i,n_i,f,k}\right), \forall\, i \in I \tag{6}$$

$$B_{i,j} \geqslant 0, \forall\, i \in I, j \in J_i \tag{7}$$

where constraint set (1) represents that each job can be assigned to only one factory. Constraint set (2) restricts that the operations of the same job must be processed in the same factory. In addition, constraint set (2) also restricts that one operation is assigned to only one machine. Constraint set (3) enforces that the operations of the same job must follow the predefined processing sequence. Constraint sets (4)–(5) ensure the nonoverlap of the operations assigned to the same machine. Constraint set (6) defines the makespan. Constraint set (7) defines that the decision variables are nonnegative.

3.4. Model 2 (position-based model)

Model 2 is based on position-based modeling idea, and it is inspired by the MILP model for DJSP (Naderi & Azab, 2014) and the MILP model for FJSP (Fattahi et al., 2007). Model 2 introduces the position decision variable $X_{i,j,f,k,p}$ to determine the machine selection and sequencing problems.

3.4.1. Decision variable

For Model 2, five decision variables namely $X_{i,j,f,k,p}$, $Z_{i,f}$, $S_{f,k,p}$, $B_{i,j}$ and $C_{\max}$ are needed.

3.4.2. Constraint sets

Constraint sets (1), and (7)–(16) are needed in Model 2.

$$Z_{i,f} = \sum_{k\in K_{i,j,f}}\sum_{p\in P_{f,k}} X_{i,j,f,k,p}, \;\; \forall\, i \in I, j \in J_i, f \in F \tag{8}$$

$$B_{i,j} + \sum_{f\in F}\sum_{k\in K_{i,j,f}}\sum_{p\in P_{f,k}} (pt_{i,j,f,k}X_{i,j,f,k,p}) \leqslant B_{i,j+1}, \forall\, i \in I, j \in JJ_i \tag{9}$$

$$\sum_{i\in I}\sum_{j\in J_i} X_{i,j,f,k,p} \geqslant \sum_{i'\in I}\sum_{j'\in J_{i'}} X_{i',j',f,k,p+1}, \forall\, f \in F, k \in K_f, p \in PP_{f,k} \tag{10}$$

$$\sum_{i\in I}\sum_{j\in J_i} X_{i,j,f,k,p} \leqslant 1, \forall\, f \in F, k \in K_f, p \in P_{f,k} \tag{11}$$

$$S_{f,k,p+1} \geqslant S_{f,k,p} + \sum_{i\in I}\sum_{j\in J_i} (pt_{i,j,f,k}X_{i,j,f,k,p}), \forall\, f \in F, k \in K_f, p \in PP_{f,k} \tag{12}$$

$$S_{f,k,p} \geqslant B_{i,j} - M(1 - X_{i,j,f,k,p}), \forall\, i \in I, j \in J_i, f \in F, k \in K_{i,j,f}, p \in P_{f,k} \tag{13}$$

Table 1
Notations and decision variables.

Notations	
i, i'	indexes for jobs
n	total number of jobs
I	set for jobs, where $I = \{1, 2, \cdots, n\}$
j, j'	index for operations
n_i	number of operations of job i
$n_{\max}$	maximum number of operations for all jobs
J_i	set for the operations of job i, where $J_i = \{1, 2, \cdots, n_i\}$
JJ_i	set for partial operations of job i, where $J_i = \{1, 2, \cdots, n_i - 1\}$
k, k'	indexes for machines
$O_{i,j}$	the j-th operation of job i
f	index for factory
r_f	number of factory
F	set of factories, where $F = \{1, \ldots, r_f\}$
m_f	number of machines in factory f
$m_{i,j,f}$	number of machines in factory f eligible for processing $O_{i,j}$
K_f	set for machines in factory f, where $K_f = \{1, 2, \cdots, m_f\}$
$K_{i,j,f}$	set for machines in factory f that are eligible to process $O_{i,j}$
$x_{i,j,f,k}$	binary constant that is equal to 1 if operation $O_{i,j}$ can be assigned to machine k in factory f, otherwise it is equal to 0
$pt_{i,j,f,k}$	processing time for operation $O_{i,j}$ processed by machine k in factory f
M	a large positive number
p	index for position
$p_{f,k}$	number of positions of machine k in factory f, where $p_{f,k} = \sum_{i \in I} \sum_{j \in J_i} x_{i,j,f,k}$
$P_{f,k}$	set of positions of machine k in factory f, where $P_{f,k} = \{1, 2, \cdots, p_{f,k}\}$
$PP_{f,k}$	set of partial positions for machine k in factory f, where $PP_{f,k} = \{1, 2, \cdots, p_{f,k} - 1\}$
t	index for discrete time
$t_{\max}$	max planned time in consideration, which should be decided in advance
T	the planning horizon, where $T = \{1, \ldots, t_{\max}\}$
$Tr_{i,f}$	time for traveling job i from factory f to the final warehouse (shipping destination)
Binary decision variables	
$X_{i,j,f,k}$	$X_{i,j,f,k} = \begin{cases} 1, & \text{if operation } O_{i,j} \text{ is assigned on machine } k \text{ of factory } f \\ 0, & \text{otherwise} \end{cases}$
$X_{i,j,f,k,p}$	$X_{i,j,f,k,p} = \begin{cases} 1, & \text{if operation } O_{i,j} \text{ is assigned to position } p \text{ of machine } k \text{ in factory } f \\ 0, & \text{otherwise} \end{cases}$
$Y_{i,j,i',j'}$	$Y_{i,j,i',j'} = \begin{cases} 1, & \text{if operation } O_{i,j} \text{ is processed before operation } O_{i',j'} \\ 0, & \text{otherwise} \end{cases}, i < i'$
$Y_{i,j,f,k,t}$	$Y_{i,j,f,k,t} = \begin{cases} 1, & \text{if at time } t\text{, operation } O_{i,j} \text{ is processed on machine } k \text{ in factory } f \\ 0, & \text{otherwise} \end{cases}$
$Y_{i',j',i,j,f,k}$	$Y_{i',j',i,j,f,k} = \begin{cases} 1, & \text{if operation } O_{i,j} \text{ is processed immediately after operation } O_{i',j'} \text{ on machine } k \text{ in factory } f \\ 0, & \text{otherwise} \end{cases}$
$A_{i,j,f,k,t}$	$A_{i,j,f,k,t} = \begin{cases} 1, & \text{if operation } O_{i,j} \text{ begins processed on machine } k \text{ in factory } f \text{ at time } t \\ 0, & \text{otherwise} \end{cases}$
$Z_{i,f}$	$Z_{i,f} = \begin{cases} 1, & \text{if job } i \text{ is assigned to factory } f \\ 0, & \text{otherwise} \end{cases}$
Continuous decision variables	
$B_{i,j}$	continuous decision variable, which defines the starting time of operation $O_{i,j}$
$S_{f,k,p}$	continuous decision variable, which defines the starting time of position p of machine k in factory f
$C_{\max}$	makespan

$$S_{f,k,p} \leqslant B_{i,j} + M(1 - X_{i,j,f,k,p}), \forall\, i \in I, j \in J, f \in F,\, k \in K_{i,j,f},\, p \in P_{f,k} \tag{14}$$

$$C_{\max} \geqslant B_{i,n_i} + \sum_{f \in F} \sum_{k \in K_{i,n_i,f}} \sum_{p \in P_{f,k}} \left(pt_{i,n_i,f,k} X_{i,n_i,f,k,p}\right), \forall\, i \in I \tag{15}$$

$$S_{f,k,p} \geqslant 0, \forall\, f \in F,\, k \in K_f,\, p \in P_{f,k} \tag{16}$$

where constraint sets (8) and (9) do the same as what constraint sets (2) and (3) do respectively. Constraint set (10) forces that the positions of each machine in each factory must be assigned to operations in sequential order. Constraint set (11) indicates that each position of each machine in each factory can be allocated at no more than one operation. Constraint set (12) does the same as what constraint sets (4)–(5) together do and ensures the nonoverlap of the operations assigned to the same machine. Constraint sets (13)–(14) represent the relationship between variables $S_{f,k,p}$ and $B_{i,j}$. Constraint set (15) determines the makespan and constraint set (16) defines the decision variable $S_{f,k,p}$.

3.5. Model 3 (time-indexed model)

Model 3 is based on time-indexed modeling idea, and it is inspired by the MILP model for OSP (Matta, 2009) and the MILP model for HFSP (Schulz, Neufeld, & Buscher, 2019). This model introduces two time-indexed binary decision variables namely $Y_{i,j,f,k,t}$ and $A_{i,j,f,k,t}$ to determine machine selection problem and sequencing problem. For this model, the max planned time $t_{\max}$ should be decided reasonably in advance. This is the number of decision variables and the number of constraints of Model 3 are heavenly dependent on $t_{\max}$.

3.5.1. Decision variable

Five decision variables namely $Y_{i,j,f,k,t}$, $Z_{i,f}$, $A_{i,j,f,k,t}$, $B_{i,j}$ and $C_{\max}$ are needed in Model 3.

3.5.2. Constraint sets

Constraint sets (1) and (17)–(23) are needed in Model 3.

$$\sum_{k\in K_{i,j,f}}\sum_{t\in T} A_{i,j,f,k,t} = Z_{i,f},\ \forall\ i\in I, j\in J_i, f\in F \tag{17}$$

$$B_{i,j} + \sum_{f\in F}\sum_{k\in K_{i,j,f}}\sum_{t\in T}(pt_{i,j,f,k}A_{i,j,f,k,t}) \leqslant B_{i,j+1},\ \forall\ i\in I, j\in JJ_i \tag{18}$$

$$\sum_{t\in T} Y_{i,j,f,k,t} = \sum_{t\in T} A_{i,j,f,k,t}pt_{i,j,f,k},\ \forall\ i\in I, j\in J_i, f\in F,\ k\in K_{i,j,f} \tag{19}$$

$$Y_{i,j,f,k,t} - Y_{i,j,f,k,t-1} \leqslant A_{i,j,f,k,t},\ \forall\ i\in I, j\in J_i, f\in F,\ k\in K_{i,j,f},\ t \in \{2,\ \ldots,t_{\max}\} \tag{20}$$

$$Y_{i,j,f,k,1} = A_{i,j,f,k,1},\ \forall\ i\in I, j\in J_i, f\in F,\ k\in K_{i,j,f} \tag{21}$$

$$B_{i,j} = \sum_{f\in F}\sum_{t\in T}\sum_{k\in K_{i,j,f}} A_{i,j,f,k,t}t,\ \forall\ i\in I, j\in J_i \tag{22}$$

$$\sum_{i\in I}\sum_{j\in J_i} Y_{i,j,f,k,t} \leqslant 1,\ \forall f\in F,\ k\in K_f,\ t\in T \tag{23}$$

$$C_{\max} \geqslant B_{i,n_i} + \sum_{f\in F}\sum_{k\in K_{i,n_i,f}}\sum_{t\in T}\left(pt_{i,n_i,f,k}A_{i,n_i,f,k,t}\right),\ \forall\ i\in I \tag{24}$$

where constraint sets (17) and (18) do the same as what constraint sets (2) and (3) do respectively. Constraint set (19) indicates that if the machine is selected to process a certain operation, the total time taken to machine the operation must equal to its processing time on the machine. Constraint sets (20)–(21) describe the relationships among decision variables $Y_{i,j,f,k,t}$ and $A_{i,j,f,k,t}$, and they enforces no interruption for each operation. Constraint set (22) shows the relationship between $B_{i,j}$ and $A_{i,j,f,k,t}$. Constraint set (23) ensures that each machine can at most process one operation at the same time, and constraint set (24) determines the makespan.

3.6. Model 4 (adjacent sequence-based model)

Model 4 is based on adjacent sequence-based modeling idea, and it is inspired by the MILP model for DJSP (Naderi & Azab, 2014) and the MILP model for FJSP (Mousakhani, 2013). This model defines the adjacent precedence decision variable $Y_{i',j',i,j,f,k}$, which decades the precedence relationship between two adjacent operations on the same machine. Moreover, a dummy job 0 must be added to aid with defining and identifying the first job to be processed on a machine.

3.6.1. Decision variable

Four decision variables namely $Y_{i',j',i,j,f,k}$, $Z_{i,f}$, $B_{i,j}$ and $C_{\max}$ are needed in Model 4.

3.6.2. Constraint sets

Model 4 is subjected to constraint sets (1), (7), and (25)–(31).

$$\sum_{i'\in\{0\cup I\}}\sum_{j'\in J_{i'}}\sum_{k\in K_{i',j',f}} Y_{i',j',i,j,f,k} = Z_{i,f},\ \forall\ i\in I, j\in J_i, f\in F \tag{25}$$

$$\sum_{i'\in\{0\cup I\}}\sum_{j'\in J_{i'}} Y_{i,j,i',j',f,k} = \sum_{i'\in\{0\cup I\}}\sum_{j'\in J_{i'}} Y_{i',j',i,j,f,k},\ \forall\ i\in\{0\cup I\}, j\in J_i, f\in F,\ k \in K_{i,j,f} \tag{26}$$

$$\sum_{i'\in\{0\cup I\}}\sum_{j'\in J_{i'}} Y_{i,j,i',j',f,k} \leqslant 1,\ \forall\ i\in\{0\cup I\}, j\in J_i, f\in F,\ k\in K_{i,j,f} \tag{27}$$

$$B_{i,j} + \sum_{f\in F}\sum_{k\in K_{i',j',f}}\sum_{i'\in\{0\cup I\}}\sum_{j'\in J_{i'}} Y_{i',j',i,j,f,k}pt_{i,j,f,k} \leqslant B_{i,j+1},\ \forall\ i\in I, j\in JJ_i \tag{28}$$

$$B_{i,j} \geqslant B_{i',j'} + \sum_{f\in F}\sum_{k\in K_{i,j,f}}\left(Y_{i',j',i,j,f,k}pt_{i',j',f,k}\right) - M\Bigg(1 - \sum_{f\in F}\sum_{k\in K_f} Y_{i',j',i,j,f,k}\Bigg),\ \forall\ i\in I, j\in J_i,\ i'\in\{0\cup I\},\ j'\in J_{i'} \tag{29}$$

$$C_{\max} \geqslant B_{i,n_i} + \sum_{f\in F}\sum_{k\in K_{i',j',f}}\sum_{i'\in\{0\cup I\}}\sum_{j'\in J_{i'}}(Y_{i',j',i,n_i,f,k}pt_{i,n_i,f,k}),\ \forall\ i\in I \tag{30}$$

$$B_{0,j} = 0,\ \forall\ j\in J_0 \tag{31}$$

where constraint set (25) has the same effect as constraint set (2). Constraint set (26) shows that if an operation is assigned to a machine in a factory, it has both the successor and predecessor on the same machine. Constraint set (27) ensures that each operation has at most one succeeding operation. Constraint set (28) does the same as what constraint set (3) does. Constraint set (29) ensures the nonoverlap of the operations assigned to the same machine. Constraint set (30) determines the makespan, and constraint set (31) defines that the operations of dummy job begin at time 0.

3.7. Lower bounds

For DFJSP, De Giovanni and Pezzella (2010) proposed a lower bound, which is calculated as,

$$C_{\max} \geqslant LB = \max_{i\in I}\{min_{f\in F}\Big\{\sum_{j\in J_i}\min_{k\in K_{i,j,f}}\{pt_{i,j,f,k}\}\Big\}\Big\} \tag{32}$$

Moreover, the makespan $C_{\max}$ is no less than the sum of maximum machine load of all factories and minimum transportation time of all jobs from all factories, therefore, the following equation holds,

$$C_{\max} \geqslant \max_{f\in F,k\in K_f}\Big\{\sum_{i\in I}\sum_{j\in J}\{X_{i,j,f,k}pt_{i,j,f,k}\}\Big\} \tag{33}$$

$$C_{\max} \geqslant \max_{f\in F,k\in K_f}\Big\{\sum_{i\in I}\sum_{j\in J}\sum_{p\in P_{f,k}}\{X_{i,j,f,k,p}pt_{i,j,f,k}\}\Big\} \tag{34}$$

$$C_{\max} \geqslant \max_{f\in F,k\in K_f}\Big\{\sum_{i\in I}\sum_{j\in J}\sum_{t\in T}\{A_{i,j,f,k,t}pt_{i,j,f,k}\}\Big\} \tag{35}$$

$$C_{\max} \geqslant \max_{f\in F,k\in K_f}\Big\{\sum_{i'\in\{0\cup I\}}\sum_{j'\in J_{i'}}\sum_{i\in I}\sum_{j\in J_i}\big\{Y_{i',j',i,j,f,k}pt_{i,j,f,k}\big\}\Big\} \tag{36}$$

Experiments show that by adding the above lower bound can improve the convergence efficiency of the MILP model, therefore, the final MILP models used to solve DFJSP are coupled with these lower bounds. In detail, the final Model 1 is coupled with constraint sets (32)–(33), the final Model 2 is coupled with constraint sets (32) and (34), the final Model 3 is coupled with constraint sets (32) and (35), and the final Model 4 is coupled with constraint sets (32) and (36).

4. CP model

Constraint programming is an exact solution method and well-known for its strength to find good quality feasible solutions for large-sized combinatorial optimization problems within reasonable time. In order to convey information between constraints and variables, CP works by using constraint propagation (filtering) method. Each global constraint is associated with a propagation algorithm that is used to remove the values of variables from their domains in order to prevent constraints from being infeasible (Gedik et al., 2016). In CP, new decision variables named interval decision variable and sequence decision variable are two popularly used variables.

An interval variable represents for a time interval, during which a task takes place and whose position in time is unknown. Moreover, it has several attributes such as start, end and processing times. In this study, we utilize interval decision variables to represent the start and end time of operations on their corresponding machines. The interval variables can be optional; that is to say, it can or cannot be present in the final solution schedule. If an interval variable is present in the final solution, it must be processed and numerical values must be assigned to its start, end and processing times respectively. If otherwise, then the domain of the interval variable is left empty (IBM, 2014).

With regard to a sequence decision variable, it represents for an ordering over a set of interval variables. For example, if a sequence is defined over a set of interval variables {A1, A2, A3, A4}, a sequence solution can be (A1, A3, A4, A2). Moreover, each interval variable in a sequence is associated with an attribute that reflects one of its properties or a combination of properties that can be employed for sequencing decisions. To avoid overlapping, related constraints must be added to the interval variables belonging to the sequence (IBM, 2014).

Here we show the CP model of DFJSP coded in IBM CPLEX Studio IDE 12.7.1. The followings represent the used parameters, decision variables and related functions in the CP model. As far as we know, there is no standardization in defining variables, related functions and constraints in the CP community. In different solvers such as CPLEX, OR Tools and Gecode, features of the same meaning may use different names. Therefore, a CP model coded in one solver may not be run successfully in another one (Gedik, Kirac, Bennet Milburn, & Rainwater, 2017). **The following model can further be understood by referring to the appendix, which shows the detailed CP model code in IBM CPLEX Studio IDE 12.7.1.**

Parameters:	
$opId$	The step of an operation in all the operations of all jobs.
$Ops_{opId,i,j}$	It refers to operation $O_{i,j}$.
$Modes_{opId,f,k,pt}$	It refers to eligible factories, corresponding machines and processing times of operation $O_{i,j}$. $Ops_{opId,i,j}$ corresponds to no less than one $Modes_{opId,f,k,pt}$ with the same $opId$ due to the flexible factories and machines.
Decision variables:	
$ops_{i,j}$	interval object of $Ops_{opId,i,j}$.
$modes_{opId}$ (optional)	optional interval object of $Modes_{opId,f,k,pt}$.
$mchs_k$	sequence decision variable, which includes a set of interval variables $modes_{opId}$ that can be assigned to machine k.
$C_{\max}$	makespan
Functions:	
$presenceOf(.)$	This function returns the presence status(Boolean type) of an interval variable. If an interval variable is present then 1 is returned; otherwise, 0 is returned.
$endOf(.)$	This function returns the end time of an interval variable if it is present; otherwise, 0 is returned by default.
$endBeforeStart(.)$	This function constrains that if predecessor and successor of an interval variable is present, then the successor cannot start before the end of the predecessor.
$alternative(.)$	$alternative(ops,\ modes)$ formulates the alternative constraint between interval variable ops and optional interval variable $modes$.
$noOverlap(.)$	This function is used to constraint the non-overlapping of the intervals present in a sequence variable. For example, a set of interval variables $\{a_1, ..., a_n\}$ are present in a sequence. Then, whenever both interval variables a_i and a_j, $i! = j$, a_i must end before the start of a_j or a_j must end before the start of a_i.

$$alternative(ops_{i,j},\ modes_{opId}):\ O_{i,j} \to opId,\ i \in I,\ j \in J_i \tag{37}$$

$$noOverlap(mchs_k):\ k \in K \tag{38}$$

$$endBeforeStart(ops_{i,j},\ ops_{i,j+1}):\ i \in I,\ j \in \{1, ..., n_i - 1\} \tag{39}$$

$$\sum_{k_2 \in K_{i,j,f_2}} presenceOf(modes_{opId_2}) \times f_2 = \sum_{k_1 \in K_{i,j',f_1}} presenceOf(modes_{opId_1}) \times f_1 : O_{i,j} \to opId_1,\ O_{i,j'} \to opId_2,\ i \in I,\ j \in J_i,\ j' \in J_i,\ f_1 \in F,\ f_2 \in F \tag{40}$$

$$C_{\max} \geqslant endOf(modes_{opId}):\ i \in I,\ O_{i,n_i} \to opId,\ f \in F \tag{41}$$

where constraint set (37) ensures that each operation is assigned to exactly one factory and one of its eligible machines. In detail, the function $alternative(.)$ is used to ensure that just one of the optional interval variables $Modes_{opId,f,k,pt}$ with the same $opId$ will be present in the final scheduling solution. Constraint set (38) assures the non-overlapping of the operations assigned to the same machine. To be specific, the function $noOverlap(.)$ enforces that all the optional interval variables $Modes_{opId,f,k,pt}$ associated with sequence variable $mchs_k$ do not overlap with each other. Constraint set (39) forces the precedence relation between the operations of the same job. Constraint set (40) ensures that the operations of the same job must be assigned to the same factory. Constraint set (41) states that the makespan is no less than the sum of the completion of each job and its corresponding transportation time.

5. Experimental results

This section evaluates the performance of the four MILP models and CP model. To compare the four MILP models, 20 instances adapted from MFJS01-10 (Fattahi et al., 2007) and MK01-MK10 (Brandimarte, 1993) are used. Each one of the 20 instances has been obtained by supposing that all the factories share the same characteristics and by replicating the data describing the related flexible job shop for each factory. Cases with only two factories are considered. Moreover, to compare the best MILP model, the CP model with state-of-the-art algorithms for solving DFJSP, three experiments involving 23 instances (rdata) with 2–4 factories from reference (De Giovanni & Pezzella, 2010) are conducted. All the MILP models and CP model are coded in OPL language of IBM CPLEX Studio IDE 12.7.1 on a desktop Dell Vostro 3900 with 3.20 GHz Intel Corei5-4460 Duo processor and 8 GB of RAM memory. The MILP models and CP model are solved by IBM CPLEX Solver and IBM CP Solver respectively.

The default method of IBM CPLEX Solver for solving MILP is the branch-and-cut method, which is the combination of cutting plane and branch-and-bound methods. The default method of IBM CP Solver for solving CP model works by using constraint propagation (filtering) method. The timelimit of all the MILP models and CP model for all instances are set to 600 s. The *RelativeOptimalityTolerance* and *OptimalityTolerance* of MILP and CP models is set to 0 to guarantee to obtain optimal solution. For CP model, the *RestartFailLimit* parameter is an important factor, which determines when the search in the CP tree needs to be restarted (Gedik et al., 2018). By try and trail, we set *RestartFailLimit* to 1000.

5.1. Comparisons of different MILP models

The four MILP models are compared from two aspects namely size complexity and computational complexity. With regard to the comparison of size complexity, three norms namely the number of binary variables (NBVs), the number of continuous variables (NCVs), and the number of constraints (NCs) are used. As to the computational complexity, comparison norms such as the final solution value (CS), the CPU time and gap value are utilized, among which the gap value represents the average optimality gap of the obtained solution within timelimit. Gap can be calculated as $|CS - BS|/|CS|\%$. Thereinto, BS is the obtained lower bound within the timelimit by the MILP model.

Table 2
Size complexity of the four proposed MILP models.

Inst.	Model 1			Model 2			Model 3			Model 4		
	NBVs	NCs	NCVs	NBVs	NCs	NCVs	NBVs	NCs	NCVs	NBVs	NCs	NCVs
MFJS01	**127**	**370**	**16**	380	1060	82	54,292	32,224	**16**	526	478	17
MFJS02	**149**	**448**	**16**	472	1290	94	62,566	37,072	**16**	644	508	17
MFJS03	**203**	675	**19**	716	1859	115	77,004	44,325	**19**	924	**659**	20
MFJS04	**254**	898	**22**	946	2392	134	112,238	63,374	**22**	1186	**824**	23
MFJS05	**257**	874	**22**	924	2340	132	92,634	52,450	**22**	1160	**820**	23
MFJS06	**312**	1117	**25**	1176	2909	149	154,024	85,975	**25**	1440	**999**	26
MFJS07	**458**	1765	**33**	1892	4501	189	240,568	131,427	**16**	2220	**1551**	34
MFJS08	**519**	1931	**37**	2034	4860	209	265,242	145,346	**37**	2396	**1882**	38
MFJS09	**719**	2831	**45**	2948	6850	251	317,674	171,640	**45**	3378	**2632**	46
MFJS10	**847**	3407	**49**	3540	8119	273	426,072	228,759	**49**	4006	**3057**	50
MK01	**1140**	4946	**56**	5266	11,633	286	14,280	8067	**56**	5740	**3811**	57
MK02	**1890**	17,333	**59**	19,248	40,609	539	20,012	11,077	**59**	40,042	**4657**	60
MK03	**8492**	49,013	**151**	51,888	107,926	1053	200,274	103,704	**151**	53,710	**25,122**	152
MK04	**2095**	**8681**	**91**	9094	19,866	435	28,238	15,648	**91**	9800	9306	92
MK05	**3734**	16,125	**107**	16,876	35,574	469	73,154	38,364	**107**	17,610	**12,534**	108
MK06	**8875**	50,273	**151**	55,020	114,513	1131	139,180	72,893	**151**	57,002	**25,285**	152
MK07	**4549**	31,005	**101**	32,350	67,297	667	91,732	47,815	**101**	33,494	**11,687**	102
MK08	**6400**	**25,916**	**226**	26,428	56,259	870	387,728	201,861	**226**	27,738	53,123	227
MK09	**15,296**	**15,296**	**241**	79,920	165,571	1453	972,064	496,687	**241**	82,366	61,309	242
MK10	**19,168**	101,841	**241**	106,504	219,617	1673	1,578,104	804,031	**241**	109,392	**61,755**	242

Values in bold indicate that they are the best.

Table 3
Computational complexity of the four proposed MILP models.

Inst.	LB	Model 1			Model 2			Model 3				Model 4		
		CS	Time	Gap	CS	Time	Gap	$t_{\max}$	CS	Time	Gap	CS	Time	Gap
MFJS01	403	**403**	0.06	0	**403**	0.34	0	410	–	600	–	**403**	0.14	0
MFJS02	396	**396**	0.03	0	**396**	0.66	0	400	–	600	–	**396**	0.17	0
MFJS03	396	**396**	0.16	0	**396**	7.15	0	400	–	600	–	**396**	0.66	0
MFJS04	496	**496**	0.25	0	**496**	7.58	0	500	–	600	–	**496**	0.56	0
MFJS05	414	**414**	0.22	0	**414**	10.30	0	420	–	600	–	**414**	0.38	0
MFJS06	614	**614**	0.19	0	**614**	10.45	0	620	–	600	–	**614**	0.89	0
MFJS07	764	**764**	0.55	0	**764**	144.36	0	770	–	600	–	**764**	56.18	0
MFJS08	764	**764**	0.44	0	**764**	167.03	0	770	–	600	–	**764**	4.57	0
MFJS09	764	**764**	3.68	0	789*	600	3.17	770	–	600	–	**764**	111.31	0
MFJS10	944	**944**	4.10	0	1222*	600	22.75	950	–	600	–	**944**	180.26	0
MK01	22	**24**	8.97	0	60	600	63.33	30	**24**	388.83	0	29	600	24.13
MK02	18	**19**	162.60	0	–	600	–	20	**19**	485.85	0	–	600	–
MK03	63	105*	600	2.86	–	600	–	110	–	600	–	–	600	–
MK04	35	**39**	513.57	0	–	600	–	40	–	600	–	199	600	81.91
MK05	59	93*	600	7.53	–	600	–	100	–	600	–	–	600	–
MK06	33	67*	600	46.27	–	600	–	70	–	600	–	–	600	–
MK07	44	75*	600	8.00	–	600	–	80	–	600	–	–	600	–
MK08	162	278*	600	5.76	–	600	–	300	–	600	–	–	600	–
MK09	130	379*	600	59.90	–	600	–	400	–	600	–	–	600	–
MK10	113	503*	600	77.53	–	600	–	550	–	600	–	–	600	–
Mean							–				–			–

–No feasible solution is found within 600 s.
Values in bold indicate that they are the best.
In this paper, $t_{\max}$ of Model 3 is determined by referring the CS of Model 1.
* Feasible but not optimum solution.

Obviously, a solution with the gap value of 0 is optimal. CS is the obtained solution within the timelimit. The size complexity and computational complexity of the four proposed MILP models for solving 20 instances are given in Tables 2 and 3.

5.1.1. Size complexity comparison

As shown in Table 2, Model 1 has much less NBVs than the other models and Model 3 has much more NBVs than the other models. Moreover, the order of the least to most of NBVs is Model 1, Model 2, Model 4 and Model 3. This is because Model 1 is based on sequence-based modeling idea, which defines the precedence decision variable $Y_{i,j,i',j'} (i < i')$ with much less binary variables than the decision variables for sequencing in other models. Model 3 is based on the time-indexed modeling idea, and $Y_{i,j,f,k,t}$ and $A_{i,j,f,k,t}$ are directly proportional to the time horizon and instance size, especially the latter. For example, although the size of MK01 is much bigger than MFJS01, the NBVs of the former is much less than the latter. This is because the time horizon of MFJS01 is 410, which is much bigger than 30 of MK01.

With regard to NCs, Model 1 is the least one for MFJS01-02, MK04 and MK08-09. Model 4 has the least NCs for MFJS03- MFJS10, MK01-MFJS03, MK05- MFJS07 and MK10. Like NBVs, the NCs of Model 3 depends most on the time horizon. Model 3 has the most NCs with regard to instances namely MFJS01- MFJS10 and MK05- MK10 with relatively large time horizon.

When it comes to NCVs, Model 1 and Model 3 have the least NCVs. This is because they only have two continuous decision variables namely $B_{i,j}$ and $C_{\max}$. Model 2 needs the most NCVs because it possesses three continuous decision variables namely $S_{f,k,p}$, $B_{i,j}$ and $C_{\max}$.

5.1.2. Computational complexity comparison

Table 3 shows the computational complexity of the four proposed MILP models. As can be seen from Table 3, Model 1 performs best in terms of CS, CPU time and gap value, which can attribute to its least NBVs. This is because NBVs is the most important influencing factor for MILP models. In detail, Model 1 can solve 13 instances out of 20 to optimality with gap = 0. Model 2 can solve 8 small instances namely MFJS01-08 to optimality with gap = 0 within 600 s. With regard to MFJS09-MFJS10 and MK01, Model 2 can only obtain feasible solutions within 600 s. For MK02-MK10, no feasible solutions can be obtained by Model 2. For Model 3, it cannot find any feasible solutions for small instances namely MFJS01- MFJS10. This is because these instances are with big time horizon, and the NBVs and NCs are very big for Model 3. Model 3 can solve relatively large-sized instances namely MK01-MK02 to optimality and outperforms Model 2 and Model 4, which can be attributed to the small time horizon of MK01- MK02. For Model 4, it can solve MFJS01-MFJS10 to optimality within 600 s, and obtain feasible solution 29 for MK01, 199 for MK04 within 600 s. For MK02-MK03 and MK05-MK10, no feasible solution can be found by Model 4. In terms of CS, CPU time and gap value, Model 4 performs better than Model 2. This may be because Model 4 has smaller NCs and NCVs than Model 2.

Above all, Model 1 outperforms Models 2–4 in terms of CS, Time and Gap, and it is most suitable for solving DFJSP. Model 4 outperforms Model 2. The performance of Model 3 is hugely dependent on the time horizon of the problem.

5.2. Comparisons of the MILP model and CP model with state-of-the- art algorithms

In this section, we evaluate the performance of the best MILP model, that is Model 1, and CP model. We compare Model 1 and CP model with three state-of-the-art algorithms for solving DFJSP, namely IGA (De Giovanni & Pezzella, 2010), GA_JS (Lu et al., 2015) and GA_OP (Wu et al., 2017). Tables 4–6 show the comparison results of 23 instances of 2–4 factories respectively. In Tables 4–6, the CPU time (T) of IGA is on a 2.0 GHz Intel Core2 processor with 2.0 GB of RAM memory in C + + (De Giovanni & Pezzella, 2010). The CPU times of GA_JS and GA_OP are on a 3.0 GHz AMD Athlon (TM) II ×4640 processor with 4.0 GB of RAM memory in C + + (Wu et al., 2017). RPE in Tables 4–6 represents for the relative percent error of the best makespan (MK) to the lower bound (LB). AV denotes the average makespan of several repetitions. In Tables 4–6, the solution with "*" denotes that it is better than the current best solution. The solution in bold is the best one among all algorithms. In Tables 4–5, for CP model, the MK in parentheses corresponds to the T(s) in parentheses. The MK in parentheses is the earliest good solution that CP model can obtain, and the T(s) in parentheses is the earliest time for obtaining the earliest good solution. For example, with regard to la06 in Table 4, CP model can obtain improved solution of 413 with 35.9 s, and it can obtain the earliest 'improved solution of 420 with 3.8 s. Between 3.8 s and 35.9 s, CP model improves solution of 420 to 413. With regard to la11 in Table 4, CP model can obtain solution of 545 within 600 s, and it can obtain earliest 545 with 12.2 s. Between 12.2 s and 600 s, CP model cannot improve the solution of 545.

5.2.1. Comparison of MILP model with the state-of-the- art algorithms

Tables 4–6 show the results of the best MILP Model 1 for all the 23 instances. In Table 4, for la01-la05, la16-la20 mt06 and mt10, Model 1 could obtain the best (optimal) solution more efficiently than all the meta-heuristic algorithms. For la10, Model 1 can obtain the optimal solution of 443 with 27.6 s, which is smaller than that of IGA and bigger than that of GA_JS and GA_OP. For la06, Model 1 can obtain very good solution of 418, which is better than 445 of IGA, 421 of GA_JS and 424 of GA_OP. For la07, in terms of MK, Model 1 performs better than IGA and GA_JS, and performs the same with GA_OP. For la09, Model 1 obtains the solution of 449, which is better than 469 for IGA, but worse than 447 for GA_JS and 444 for GA_OP. For other instances namely la11-la15 and mt20, the comparison results are obviously shown in Table 4.

For the instances with 3 factories in Table 5, in terms of MK, Model 1 performs no worse than all the existing algorithms for all the instances

Table 4
Comparison results of 2-factory DFJSP.

Inst.	LB	IGA				GA_JS				GA_OP				CP model			Model 1		
		MK	AV	T(s)	RPE	MK	AV	T(s)	RPE	MK	AV	T(s)	RPE	MK	T(s)	RPE	MK	T(s)	RPE
la01	413	**413**	413	12	0	**413**	413	7.1	0	**413**	413	4.7	0	**413**	0.1	0	**413**	3.2	0
la02	394	**394**	394	11.2	0	**394**	394	6.6	0	**394**	394	4.7	0	**394**	0.1	0	**394**	1.3	0
la03	349	**349**	349	10.8	0	**349**	349	7	0	**349**	349	5.2	0	**349**	0.1	0	**349**	4.9	0
la04	369	**369**	369	11.4	0	**369**	369	6.4	0	**369**	369	5	0	**369**	0.1	0	**369**	1.0	0
la05	380	**380**	380	8	0	**380**	380	6.6	0	**380**	380	5	0	**380**	0.1	0	**380**	0.9	0
la06	413	445	449.6	45.8	7.7	421	435.8	25.8	1.9	424	432.7	23.9	2.7	**413***(420*)	35.9(3.8)	0(1.7)	418*	600	1.2
la07	376	412	419.2	50.2	9.6	396	408.5	26.3	5.3	390	403.6	24.6	3.7	**386***(389*)	177.2(25.2)	2.7(3.5)	390	600	3.7
la08	369	420	427.8	53.8	13.8	406	417.4	26.4	10	397	411.7	23.7	7.6	**391***(396*)	82.8(11.9)	6.0(7.3)	399	600	8.1
la09	382	469	474.6	45.2	22.8	447	459	27	17	444	455.7	24.1	16.2	**436***(443*)	351.5(10.4)	14.1(16.0)	449	600	17.5
la10	443	445	448.6	45	0.5	**443**	444.1	24.9	0	**443**	443.2	21.9	0	**443**	0.1	0	**443**	27.6	0
la11	413	570	571.6	126	38	548	557.1	64.6	32.7	**541**	549.9	62.5	31	545(5 4 5)	600(12.2)	32.0	567	600	37.3
la12	408	504	508	116	23.5	483	492.5	65.5	18.4	474	482.3	63.9	16.2	**469***(473*)	600(35.8)	15.0(15.9)	485	600	18.9
la13	382	542	552.2	125.4	41.9	530	538.4	66.1	38.7	529	538.1	62.3	38.5	**525***(528*)	600(26.6)	37.4(38.2)	545	600	42.7
la14	443	570	576	122.2	28.7	545	557.3	63.3	23	544	553.7	63	22.8	**542***(5 4 4)	600(12.9)	22.3(22.8)	545	600	23.0
la15	378	584	588.8	119.6	54.5	**554**	568.7	63.6	46.6	**554**	566.6	63.1	46.6	555(5 5 5)	600(42.6)	46.8(46.8)	577	600	52.6
la16	717	**717**	717	140.2	0	**717**	717	53.3	0	**717**	717	47.8	0	**717**	0.1	0	**717**	1.9	0
la17	646	**646**	646	112.6	0	**646**	646	52	0	**646**	646	48.7	0	**646**	0.1	0	**646**	1.9	0
la18	663	**663**	663	132.4	0	**663**	663	53.9	0	**663**	663	48.7	0	**663**	0.1	0	**663**	2.7	0
la19	617	**617**	617.2	147.2	0	**617**	622.1	58.4	0	**617**	617.5	57.9	0	**617**	0.3	0	**617**	5.8	0
la20	756	**756**	756	99.8	0	**756**	756	53.9	0	**756**	756	48.3	0	**756**	0.2	0	**756**	2.4	0
mt06	47	**47**	47	2	0	**47**	47	3.4	0	**47**	47	2.4	0	**47**	0.1	0	**47**	0.1	0
mt10	655	**655**	655	173	0	**655**	655	54	0	**655**	655	49.6	0	**655**	0.1	0	**655**	2.4	0
mt20	387	560	566	121.2	44.7	530	547.7	59.5	37	525	534.4	61.4	35.7	**523***(**523***)	600(39.6)	35.1(35.1)	533	600	37.7
Ave.				79.6	12.4			38.1	10.0			35.8	9.6		184.7(9.7)	9.2(9.5)		263.3	10.6

Table 5
Comparison results of 3-factory DFJSP.

Inst.	LB	IGA				GA_JS				GA_OP				CP model			Model 1		
		MK	AV	T(s)	RPE	MK	AV	T(s)	RPE	MK	AV	T(s)	RPE	MK	T(s)	RPE	MK	T(s)	RPE
la01	413	**413**	413	4.6	0	**413**	413	4.5	0	**413**	413	3	0	**413**	0.1	0	**413**	0.9	0
la02	394	**394**	394	3.6	0	**394**	394	4.4	0	**394**	394	3	0	**394**	0.1	0	**394**	0.6	0
la03	349	**349**	349	3.8	0	**349**	349	4.6	0	**349**	349	3.1	0	**349**	0.1	0	**349**	0.7	0
la04	369	**369**	369	3.8	0	**369**	369	4.4	0	**369**	369	3	0	**369**	0.1	0	**369**	1.0	0
la05	380	**380**	380	2.6	0	**380**	380	4.4	0	**380**	380	2.9	0	**380**	0.1	0	**380**	0.5	0
la06	413	**413**	413	17.4	0	**413**	413	10.3	0	**413**	413	7.9	0	**413**	0.1	0	**413**	11.3	0
la07	376	**376**	376	18.2	0	**376**	376	11.2	0	**376**	376	9.1	0	**376**	0.1	0	**376**	10.5	0
la08	369	**369**	369	19.6	0	**369**	369	10.6	0	**369**	369	8.7	0	**369**	0.2	0	**369**	23.1	0
la09	382	**382**	387.4	17.8	0	**382**	382	12.8	0	**382**	382	11	0	**382**	0.6	0	**382**	26.0	0
la10	443	**443**	443	17	0	**443**	443	10.7	0	**443**	443	8.4	0	**443**	0.1	0	**443**	4.1	0
la11	413	425	436.8	50.6	2.9	**413**	419.3	31.2	0	**413**	418	28.9	0	**413**	1.5	0	**413**	86.7	0
la12	408	**408**	408	44.6	0	**408**	408	23.5	0	**408**	408	22.1	0	**408**	0.2	0	**408**	45.8	0
la13	382	419	430.2	45.8	9.7	396	407.6	31.9	3.7	395	408.4	29.6	3.4	**382***	3.4	0	390*	600	2.1
la14	443	**443**	448.8	48.8	0	**443**	443	23.5	0	**443**	443	23.1	0	**443**	5.4	0	**443**	106.7	0
la15	378	451	456	42.2	19.3	413	423.7	32.6	9.3	417	430	30.5	10.3	**388***(415*)	600(2.3)	2.6(9.8)	416*	600	10.1
la16	717	**717**	717	36	0	**717**	717	26.4	0	**717**	717	21.1	0	**717**	0.2	0	**717**	2.1	0
la17	646	**646**	646	31.6	0	**646**	646	25.1	0	**646**	646	21.1	0	**646**	0.1	0	**646**	1.7	0
la18	663	**663**	663	36.8	0	**663**	663	26.1	0	**663**	663	20.6	0	**663**	0.2	0	**663**	2.7	0
la19	617	**617**	617	62.4	0	**617**	617	25.8	0	**617**	617	20.2	0	**617**	0.3	0	**617**	4.3	0
la20	756	**756**	756	34.2	0	**756**	756	25.2	0	**756**	756	20.8	0	**756**	0.2	0	**756**	3.2	0
mt06	47	**47**	47	1	0	**47**	47	2.7	0	**47**	47	1.8	0	**47**	0.1	0	**47**	0.1	0
mt10	655	**655**	655	50	0	**655**	655	25.8	0	**655**	655	21.8	0	**655**	0.1	0	**655**	2.8	0
mt20	387	439	442.6	48.2	13.4	407	415.8	33	5.2	397	412.7	29.5	2.6	**387***	7.5	0	399	600	3.1
Ave.				27.9	2.0			17.9	0.8			15.3	0.7						

except for mt20. For la13, Model 1 outperforms IGA, GA_JS, and GA_OP, and obtains the best solution of 390 within 600 s. For mt20, Model 1 outperforms IGA and GA_JS, and obtains relatively good solution of 399 within 600 s. For la15, Model 1 obtains the best solution of 416. For the instances with 4 factories in Table 6, Model 1 can efficiently obtain the best and optimal solutions for all instances.

Above all, although the MILP model is time-consuming for some large-sized and hard instances, it can efficiently obtains the best or the optimal solutions for small-sized and easy instances. More importantly, Model 1 can always obtain the same solution for all the instances with the same time, and no repetitious solving is needed.

5.2.2. Comparison of CP model with the state-of-the- art algorithms

As can be seen in Tables 4–6, except for la11 and la15, CP model can obtain no worse MK and RPE for all instances than IGA, GA_JS and GA_OP. For the difficult instances, namely la06-la09, la12- la14 and mt20 with two factories in Table 4, CP model improves the best solution. For la06, CP model can obtain good solution of 420 with very short time of 3.8 s and the optimal solution of 413 with 35.9 s. For la07, CP model can obtain good solution of 389 with 25.2 s and the optimal solution of 386 with 177.2 s. For la08, CP model can quickly obtain the good solution of 396 with 11.9 s and the optimal solution of 391 with 82.8 s. With regard to la09, CP model can archive good solution of 443

Table 6
Comparison results of 4-factory DFJSP.

Inst.	LB	IGA				GA_JS				GA_OP				CP model			Model 1		
		MK	AV	T(s)	RPE	MK	AV	T(s)	RPE	MK	AV	T(s)	RPE	MK	T(s)	RPE	MK	T(s)	RPE
la01	413	**413**	413	1.8	0	**413**	413	4.4	0	**413**	413	2.4	0	**413**	0.1	0	**413**	0.4	0
la02	394	**394**	394	1.8	0	**394**	394	4.3	0	**394**	394	2.4	0	**394**	0.1	0	**394**	0.5	0
la03	349	**349**	349	2.2	0	**349**	349	4.4	0	**349**	349	2.4	0	**349**	0.1	0	**349**	0.6	0
la04	369	**369**	369	2	0	**369**	369	4.3	0	**369**	369	2.4	0	**369**	0.1	0	**369**	0.7	0
la05	380	**380**	380	1	0	**380**	380	4.4	0	**380**	380	2.4	0	**380**	0.1	0	**380**	0.7	0
la06	413	**413**	413	9	0	**413**	413	7.9	0	**413**	413	5.5	0	**413**	0.1	0	**413**	7.0	0
la07	376	**376**	376	9.6	0	**376**	376	8.3	0	**376**	376	5.7	0	**376**	0.2	0	**376**	4.0	0
la08	369	**369**	369	12.6	0	**369**	369	8.2	0	**369**	369	5.7	0	**369**	0.2	0	**369**	6.1	0
la09	382	**382**	382	11.6	0	**382**	382	8.4	0	**382**	382	5.7	0	**382**	0.2	0	**382**	13.2	0
la10	443	**443**	443	7.8	0	**443**	443	7.9	0	**443**	443	5.4	0	**443**	0.1	0	**443**	5.8	0
la11	413	**413**	413	29.6	0	**413**	413	15.7	0	**413**	413	13.4	0	**413**	0.3	0	**413**	38.7	0
la12	408	**408**	408	26.6	0	**408**	408	15.9	0	**408**	408	11.9	0	**408**	0.2	0	**408**	38.6	0
la13	382	**382**	382	27.6	0	**382**	382	17.6	0	**382**	382	15.6	0	**382**	0.3	0	**382**	18.9	0
la14	443	**443**	443	29.8	0	**443**	443	15.2	0	**443**	443	12.2	0	**443**	0.2	0	**443**	42.4	0
la15	378	**378**	397	28.8	5.0	**378**	381.9	20.3	0	**378**	385.8	17.9	0	**378**	0.8	0	**378**	104.2	0
la16	717	**717**	717	20.2	0	**717**	717	16.2	0	**717**	717	12.4	0	**717**	0.2	0	**717**	3.2	0
la17	646	**646**	646	16.4	0	**646**	646	15.2	0	**646**	646	12.2	0	**646**	0.1	0	**646**	2.9	0
la18	663	**663**	663	24.4	0	**663**	663	17.3	0	**663**	663	12.6	0	**663**	0.1	0	**663**	2.9	0
la19	617	**617**	617	33	0	**617**	617	16.1	0	**617**	617	11.3	0	**617**	0.2	0	**617**	2.6	0
la20	756	**756**	756	18	0	**756**	756	15.4	0	**756**	756	11.8	0	**756**	0.1	0	**756**	2.8	0
mt06	47	**47**	47	0.2	0	**47**	47	2.4	0	**47**	47	1.7	0	**47**	0.2	0	**47**	0.2	0
mt10	655	**655**	655	31.2	0	**655**	655	17.1	0	**655**	655	13.3	0	**655**	0.1	0	**655**	2.6	0
mt20	387	**387**	388.4	27	0	**387**	387	17.1	0	**387**	387	15.7	0	**387**	0.2	0	**387**	67.3	0
Ave.				16.2	0.2			11.5	0			8.8	0		0.2	0		15.9	0

with 10.4 s; moreover, optimal solution of 436 are obtained with 351.5 s. For la11, CP model can obtain the second-best solution of 545 with short time of 12.2 s. For la12, CP model obtains improved solution of 473 with 35.8 s. Moreover, much better solution of 469 is archived by running CP model with 600.0 s. For la13, CP model effectively obtains improved solution of 528 with 26.6 s and much better solution of 525 with 600.0 s. For la14, CP model obtains the current best solution of 544 with 12.9 s and improved solution of 542 within 600.0 s. For la15, CP model can obtain very good solution of 555 with 42.6 s. For other easy instances, CP model can solve them to optimality more efficiently than the IGA, GA_JS and GA_OP.

When it comes to the instances with 3 and 4 factories, CP model also ranks first for all instances in terms of MK and RPE for all instances. For the instances with 3 factories in Table 5, CP model can solve 22 out of 23 instances to optimality efficiently. For the difficult instance la15, CP model obtains much better solution of 415 and 388 with 2.3 s and 600.0 s respectively. For all the 23 instances with 4 factories in Table 6, CP model solves them to optimality much more efficiently with mean time of 0.2 s than all the other algorithms.

In summary, CP model improves 11 best known solutions for the benchmark instances and proves the optimality of 62 best-known solutions. Obviously, CP outperforms all the other algorithms. Moreover, the same as MILP model, CP model can always obtain the same solution for all the instances with the same time, and no repetitious solving is needed. While meta-heuristic algorithms cannot assure the same solution of each repeat even for the relatively small-sized instances.

6. Conclusions and future study

As far as we know, this paper is the first to formulate MILP models for DFJSP and is the first to implement CP method to DFJSP. For solving DFJSP, we formulate four MILP models based on four different modeling ideas. Due to the NP-hard characteristic of DFJSP, we proposed a CP model to solve large-sized instances more effectively. We compare all the MILP models and the CP model with the existing algorithms. Experimental results show that the sequence-based model is the most efficient for solving DFJSP. Moreover, the CP model outperforms the state-of-art algorithms. More importantly, the proposed MILP model and CP model improve 11 best known solutions for the benchmark instances and prove the optimality of 62 best-known solutions. Most importantly, the MILP model and CP model can always obtain the same solution for all the instances with the same time, and no repeat is needed. While meta-heuristic algorithms cannot assure the same solution of each repeat even for the relatively small-sized instances.

As we can see from the comparison results, CP model outperforms all the other existing algorithms in terms of both solution quality and efficiency. Moreover, due to the simplicity of the CP formulation and the ease of using the currently available solver IBM CP Optimizer, CP method is very suitable for practitioners to implement and use in practice.

In the future research, we will extend the four modeling ideas to other distributed scheduling problems and formulate efficient MILP models for them. The MILP models and CP model proposed in this paper can be easily modified to solve DFJSP with other objectives such total flowtime and earliness/tardiness. Moreover, we will try to formulate MILP/CP hybrid model based on decomposition methods such as logic-based benders decomposition (Gedik et al., 2016), and further merge the strengths of MILP and CP methods.

CRediT authorship contribution statement

Leilei Meng: Conceptualization, Methodology, Software, Writing - original draft. **Chaoyong Zhang:** Supervision, Writing - review & editing, Funding acquisition. **Yaping Ren:** Investigation. **Biao Zhang:** Formal analysis. **Chang Lv:** Software.

Acknowledgements

This work was supported by Project of International Cooperation and Exchanges NSFC (No. 51861165202), the Key Research and Development Program of Guangdong Province (No. 2019B090921001), and the National Natural Science Foundation of China (No. 51575211, 51705263, 51805330). We are grateful to the editor and anonymous referees for their constructive comments to improve this paper.

Appendix A

CP model coded in IBM CPLEX Studio IDE 12.7.1.

```
using CP;
int nbJobs = ...;
int nbMchs = ...;
int nbFactor = ...;
range Jobs = 1..nbJobs;
range Mchs = 1..nbMchs;
range Factors = 1..nbFactor;
tuple Operation {
  int opId; // The step of an operation in all the operations of all jobs.
  int jobId; // Job id
  int pos; // Position in job
};
tuple Mode {
  int opId; // The step of an operation in all the operations of all jobs.
  int fId; // Factory id
  int mch; // Machine
  int pt; // Processing time
};
{Operation} Ops = ...;
{Mode} Modes = ...;
// Position of last operation of job j
int jlast[j in Jobs] = max(o in Ops: o.jobId==j) o.pos;
dvar interval ops [Ops];
dvar interval modes[md in Modes] optional size md.pt;
dvar sequence mchs[m in Mchs] in all(md in Modes: md.mch == m) modes[md];
dexpr int makespan;
execute {
    cp.param.TimeLimit = 600;
```

CP model coded in IBM CPLEX Studio IDE 12.7.1.

```
        cp.param.RestartFailLimit = 1000;
        cp.param.RelativeOptimalityTolerance = 0;
        cp.param.OptimalityTolerance = 0;
}
minimize makespan;
subject to {
    forall (o in Ops)
    alternative(ops[o], all(md in Modes: md.opId==o. opId &&md.fId < =nbFactor) modes[md]);
    forall (m in Mchs)
    noOverlap(mchs[m]);
    forall (j in Jobs, o1 in Ops, o2 in Ops: o1.jobId==j && o2.jobId==j && o2.pos==1 + o1.pos)
    endBeforeStart(ops[o1],ops[o2]);
    forall(j in Jobs, o1 in Ops, o2 in Ops: o1.jobId==j&& o2.jobId==j&& o2.pos==o1.pos + 1)
    sum(md1 in Modes:o1.id==md1.opId)presenceOf(modes[md1])*md1.fId
    ==
    sum(md2 in Modes:o2.id==md2.opId)presenceOf(modes[md2])*md2.fId;
    makespan > =max(j in Jobs, o in Ops: o.pos==jlast[j]) endOf(ops[o]);
}
execute {
writeln("makespan ",makespan);
for (var m in Modes)
    {
        if (modes[m].present)
            writeln(m.opId+" " + m.fId + " " + m.mch + " " + modes[m].start + " " + modes[m].end);
    }
}
```

Appendix B. Supplementary material

Supplementary data to this article can be found online at https://doi.org/10.1016/j.cie.2020.106347.

References

Bożek, A., & Werner, F. (2018). Flexible job shop scheduling with lot streaming and sublot size optimisation. *International Journal of Production Research, 56*(19), 6391–6411.

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

Bukchin, Y., & Raviv, T. (2018). Constraint programming for solving various assembly line balancing problems. *Omega, 78*, 57–68.

Chan, F. T. S., Chung, S. H., & Chan, P. L. Y. (2006). Application of genetic algorithms with dominant genes in a distributed scheduling problem in flexible manufacturing systems. *International Journal of Production Research, 44*(3), 523–543.

Chan, F. T. S., Chung, S. H., Chan, L. Y., Finke, G., & Tiwari, M. K. (2006). Solving distributed FMS scheduling problems subject to maintenance: Genetic algorithms approach. *Robotics and Computer-Integrated Manufacturing, 22*(5–6), 493–504.

Chan, F., Chung, S., & Chan, P. (2005). An adaptive genetic algorithm with dominated genes for distributed scheduling problems. *Expert Systems with Applications, 29*(2), 364–371.

Chang, H., & Liu, T. (2017). Optimisation of distributed manufacturing flexible job shop scheduling by using hybrid genetic algorithms. *Journal of Intelligent Manufacturing, 28*(8), 1973–1986.

Chung, S. H., Chan, F. T. S., & Chan, H. K. (2009). A modified genetic algorithm approach for scheduling of perfect maintenance in distributed production scheduling. *Engineering Applications of Artificial Intelligence, 22*(7), 1005–1014.

De Giovanni, L., & Pezzella, F. (2010). An Improved Genetic Algorithm for the Distributed and Flexible Job-shop Scheduling problem. *European Journal of Operational Research, 200*(2), 395–408.

Demir, Y., & Kürşat İşleyen, S. (2013). Evaluation of mathematical models for flexible job-shop scheduling problems. *Applied Mathematical Modelling, 37*(3), 977–988.

Edis, E. B., & Oguz, C. (2012). Parallel machine scheduling with flexible resources. *Computers & Industrial Engineering, 63*(2), 433–447.

Fattahi, P., Saidi Mehrabad, M., & Jolai, F. (2007). Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal of Intelligent Manufacturing, 18*(3), 331–342.

Fleszar, K., & Hindi, K. S. (2018). Algorithms for the unrelated parallel machine scheduling problem with a resource constraint. *European Journal of Operational Research, 271*(3), 839–848.

Gedik, R., Kalathia, D., Egilmez, G., & Kirac, E. (2018). A constraint programming approach for solving unrelated parallel machine scheduling problem. *Computers & Industrial Engineering, 121*, 139–149.

Gedik, R., Kirac, E., Bennet Milburn, A., & Rainwater, C. (2017). A constraint programming approach for the team orienteering problem with time windows. *Computers & Industrial Engineering, 107*, 178–195.

Gedik, R., Rainwater, C., Nachtmann, H., & Pohl, E. A. (2016). Analysis of a parallel machine scheduling problem with sequence dependent setup times and job availability intervals. *European Journal of Operational Research, 251*(2), 640–650.

Gicquel, C., Hege, L., Minoux, M., & van Canneyt, W. (2012). A discrete time exact solution approach for a complex hybrid flow-shop scheduling problem with limited-wait constraints. *Computers & Operations Research, 39*(3), 629–636.

Ham, A. M., & Cakici, E. (2016). Flexible job shop scheduling problem with parallel batch processing machines: MIP and CP approaches. *Computers & Industrial Engineering, 102*, 160–165.

Huang, Y. J., & Yao, X. F. (2012). Planning and scheduling of multiple flexible-shops based on analytical target cascading and particle swarm optimization. *Journal of Central South University(Science and Technology), 43*(01), 151–158.

IBM (2014). IBM ILOG CPLEX Optimization Studio 12.6.

Jain, V., & Grossmann, I. E. (2001). Algorithms for hybrid MILP/CP models for a class of optimization problems. *INFORMS Journal on Computing, 13*(4), 258–276.

Jia, H. Z., Nee, A. Y. C., Fuh, J. Y. H., & Zhang, Y. F. (2003). A modified genetic algorithm for distributed scheduling problems. *Journal of Intelligent Manufacturing, 14*(3–4), 351–362.

Jin, L., Tang, Q., Zhang, C., Shao, X., & Tian, G. (2016). More MILP models for integrated process planning and scheduling. *International Journal of Production Research, 54*(14), 4387–4402.

Kelbel, J., & Hanzálek, Z. (2011). Solving production scheduling with earliness/tardiness penalties by constraint programming. *Journal of Intelligent Manufacturing, 22*(4), 553–562.

Keskinturk, T., Yildirim, M. B., & Barut, M. (2012). An ant colony optimization algorithm for load balancing in parallel machines with sequence-dependent setup times. *Computers & Operations Research, 39*(6), 1225–1235.

Ku, W., & Beck, J. C. (2016). Mixed Integer Programming models for job shop scheduling: A computational analysis. *Computers & Operations Research, 73*, 165–173.

Li, J., Bai, S., Duan, P., Sang, H., Han, Y., & Zheng, Z. (2019). An improved artificial bee colony algorithm for addressing distributed flow shop with distance coefficient in a prefabricated system. *International Journal of Production Research*, 1–21.

Li, J., Sang, H., Han, Y., Wang, C., & Gao, K. (2018). Efficient multi-objective optimization algorithm for hybrid flow shop scheduling problems with setup energy consumptions. *Journal of Cleaner Production, 181*, 584–598.

Lin, C., Lee, I., & Wu, M. (2019). Merits of using chromosome representations and shadow chromosomes in genetic algorithms for solving scheduling problems. *Robotics and Computer-Integrated Manufacturing, 58*, 196–207.

Lu, P., Wu, M., Tan, H., Peng, Y., & Chen, C. (2015). A genetic algorithm embedded with a concise chromosome representation for distributed and flexible job-shop scheduling problems. *Journal of Intelligent Manufacturing, 29*(1), 19–34.

Manne, A. S. (1960). On the Job-Shop Scheduling Problem. *Operations Research, 8*(2), 219–223.

Marzouki, B., Driss, O. B., & Ghédira, K. (2018). Solving distributed and flexible job shop scheduling problem using a chemical reaction optimization metaheuristic. *Procedia Computer Science, 126*, 1424–1433.

Matta, M. E. (2009). A genetic algorithm for the proportionate multiprocessor open shop. *Computers & Operations Research, 36*(9), 2601–2618.

Meng, L., Zhang, C., Shao, X., & Ren, Y. (2019a). MILP models for energy-aware flexible job shop scheduling problem. *Journal of Cleaner Production, 210*, 710–723.

Meng, L., Zhang, C., Shao, X., Ren, Y., & Ren, C. (2019b). Mathematical modelling and

optimisation of energy-conscious hybrid flow shop scheduling problem with unrelated parallel machines. *International Journal of Production Research, 4*(57), 1119–1145.
Meng, L., Zhang, C., Shao, X., Zhang, B., Ren, Y., & Lin, W. (2019c). More MILP models for hybrid flow shop scheduling problem and its extended problems. *International Journal of Production Research,* 1–22.
Meng, L., Zhang, C., Zhang, B., & Ren, Y. (2019d). Mathematical modeling and optimization of energy-conscious flexible job shop scheduling problem with worker flexibility. *IEEE Access, 7*(1), 68043–68059.
Mousakhani, M. (2013). Sequence-dependent setup time flexible job shop scheduling problem to minimise total tardiness. *International Journal of Production Research, 51*(12), 3476–3487.
Na, H., & Park, J. (2014). Multi-level job scheduling in a flexible job shop environment. *International Journal of Production Research, 52*(13), 3877–3887.
Naderi, B., & Azab, A. (2014). Modeling and heuristics for scheduling of distributed job shops. *Expert Systems with Applications, 41*(17), 7754–7763.
Naderi, B., Fatemi Ghomi, S. M. T., Aminnayeri, M., & Zandieh, M. (2011a). A study on open shop scheduling to minimise total tardiness. *International Journal of Production Research, 49*(15), 4657–4678.
Naderi, B., Ghomi, S. M. T. F., Aminnayeri, M., & Zandieh, M. (2011b). Scheduling open shops with parallel machines to minimize total completion time. *Journal of Computational & Applied Mathematics, 235*(5), 1275–1287.
Naderi, B., Khalili, M., & Khamseh, A. A. (2013). Mathematical models and a hunting search algorithm for the no-wait flowshop scheduling with parallel machines. *International Journal of Production Research, 52*(9), 2667–2681.
Naderi, B., Zandieh, M., & Shirazi, M. A. H. A. (2009). Modeling and scheduling a case of flexible flowshops: Total weighted tardiness minimization. *Computers & Industrial Engineering, 57*(4), 1258–1267.
Nouiri, M., Bekrar, A., Jemai, A., Niar, S., & Ammari, A. C. (2018). An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem. *Journal of Intelligent Manufacturing, 29*(3), 603–615.
Novara, F. M., Novas, J. M., & Henning, G. P. (2016). A novel constraint programming model for large-scale scheduling problems in multiproduct multistage batch plants: Limited resources and campaign-based operation. *Computers & Chemical Engineering, 93*, 101–117.
Oliveira, R. M. E. S., & Ribeiro, M. S. F. O. (2015). Comparing mixed & integer programming vs. constraint programming by solving job-shop scheduling problems. *Independent Journal of Management & Production, 6*(1).
Özgüven, C., Özbakır, L., & Yavuz, Y. (2010). Mathematical models for job-shop scheduling problems with routing and process plan flexibility. *Applied Mathematical Modelling, 34*(6), 1539–1548.
Öztürk, C., Tunalı, S., Hnich, B., & Örnek, A. (2015). Cyclic scheduling of flexible mixed model assembly lines with parallel stations. *Journal of Manufacturing Systems, 36*, 147–158.
Pan, C. (1997). A study of integer programming formulations for scheduling problems. *International Journal of Systems Science, 28*(1), 33–41.
Pan, Q., Gao, L., Wang, L., Liang, J., & Li, X. (2019). Effective heuristics and metaheuristics to minimize total flowtime for the distributed permutation flowshop problem. *Expert Systems with Applications, 124*, 309–324.
Ren, Y., Yu, D., Zhang, C., Tian, G., Meng, L., & Zhou, X. (2017). An improved gravitational search algorithm for profit-oriented partial disassembly line balancing problem. *International Journal of Production Research, 55*(24), 7302–7316.
Ren, Y., Meng, L., Zhang, C., Zhao, F., Saif, U., Huang, A., … Sutherland, J. W. (2020). An efficient metaheuristics for a sequence-dependent disassembly planning. *Journal of Cleaner Production, 245*, 118644.
Rocha, P. L., Ravetti, M. G., Mateus, G. R., & Pardalos, P. M. (2008). Exact algorithms for a scheduling problem with unrelated parallel machines and sequence and machine-dependent setup times. *Computers & Operations Research, 35*(4), 1250–1264.
Roshanaei, V., Azab, A., & ElMaraghy, H. (2013). Mathematical modelling and a metaheuristic for flexible job shop scheduling. *International Journal of Production Research, 51*(20), 6247–6274.
Ruiz, R., Pan, Q., & Naderi, B. (2019). Iterated Greedy methods for the distributed permutation flowshop scheduling problem. *Omega, 83*, 213–222.
Ruiz, R., Şerifoğlu, F. S., & Urlings, T. (2008). Modeling realistic hybrid flexible flowshop scheduling problems. *Computers and Operations Research, 35*(4), 1151–1175.
Schulz, S., Neufeld, J. S., & Buscher, U. (2019). A multi-objective iterated local search algorithm for comprehensive energy-aware hybrid flow shop scheduling. *Journal of Cleaner Production, 224*, 421–434.
Shen, L., Dauzère-Pérès, S., & Neufeld, J. S. (2018). Solving the flexible job shop scheduling problem with sequence-dependent setup times. *European Journal of Operational Research, 265*(2), 503–516.
Sun, L., Lin, L., Li, H., & Gen, M. (2019). Large scale flexible scheduling optimization by a distributed evolutionary algorithm. *Computers & Industrial Engineering, 128*, 894–904.
Vallada, E., & Ruiz, R. (2011). A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research, 211*(3), 612–622.
Wagner, H. M. (1959). An integer linear-programming model for machine scheduling. *Naval Research Logistics Quarterly, 6*(2), 131–140.
Wu, J., & Chien, C. (2008). Modeling semiconductor testing job scheduling and dynamic testing machine configuration. *Expert Systems with Applications, 35*(1–2), 485–496.
Wu, M., Lin, C., Lin, C., & Chen, C. (2017). Effects of different chromosome representations in developing genetic algorithms to solve DFJS scheduling problems. *Computers & Operations Research, 80*, 101–112.
Wu, R., Guo, S. S., Li, Y. B., & Wang, L. (2018). An improved artificial bee colony algorithm for distributed and flexible job-shop scheduling problem. *Control and Decision,* 1–11.
Wu, X., Liu, X., & Zhao, N. (2018). An improved differential evolution algorithm for solving a distributed assembly flexible job shop scheduling problem. *Memetic Computing, 6*, 1–21.
Zhang, B., Pan, Q., Gao, L., Meng, L., Li, X., & Peng, K. (2019). A three-stage multiobjective approach based on decomposition for an energy-efficient hybrid flow shop scheduling problem. *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* 1–16.
Zhang, L., Tang, Q., Wu, Z., & Wang, F. (2017). Mathematical modeling and evolutionary generation of rule sets for energy-efficient flexible job shops. *Energy, 138*, 210–227.
Ziaee, M. (2014). A heuristic algorithm for the distributed and flexible job-shop scheduling problem. *The Journal of Supercomputing, 67*(1), 69–83.