International Journal of Production Research

ISSN: 0020-7543 (Print) 1366-588X (Online) Journal homepage: https://www.tandfonline.com/loi/tprs20

A hybrid artificial bee colony algorithm for flexible job shop scheduling with worker flexibility

Guiliang Gong, Raymond Chiong, Qianwang Deng & Xuran Gong

To cite this article: Guiliang Gong, Raymond Chiong, Qianwang Deng & Xuran Gong (2019): A hybrid artificial bee colony algorithm for flexible job shop scheduling with worker flexibility, International Journal of Production Research, DOI: 10.1080/00207543.2019.1653504

To link to this article: https://doi.org/10.1080/00207543.2019.1653504

Published online: 16 Aug 2019.

International Journal of Production Research, 2019
https://doi.org/10.1080/00207543.2019.1653504

A hybrid artificial bee colony algorithm for flexible job shop scheduling with worker flexibility

Guiliang Gong[a,b], Raymond Chiong [b], Qianwang Deng[a*] and Xuran Gong[a]

[a]*State Key Laboratory of Advanced Design and Manufacturing for Vehicle Body, Hunan University, Changsha, People's Republic of China;* [b]*School of Electrical Engineering and Computing, The University of Newcastle, Callaghan, Australia*

(*Received 20 February 2019; accepted 2 July 2019*)

The traditional flexible job shop scheduling problem (FJSP) considers machine flexibility but not worker flexibility. Given the influence and potential of human factors in improving production efficiency and decreasing the cost in practical production systems, we propose a mathematical model of an extended FJSP with worker flexibility (FJSPW). A hybrid artificial bee colony algorithm (HABCA) is presented to solve the proposed FJSPW. For the HABCA, effective encoding, decoding, crossover and mutation operators are designed, and a new effective local search method is developed to improve the speed and exploitation ability of the algorithm. The Taguchi method of Design of Experiments is used to obtain the best combination of key parameters of the HABCA. Extensive computational experiments carried out to compare the HABCA with some well-performing algorithms from the literature confirm that the proposed HABCA is more effective than these algorithms, especially on large-scale FJSPW instances.

Keywords: flexible job shop scheduling; worker flexibility; human factors; single-objective optimisation; artificial bee colony algorithms

1. Introduction

The flexible job shop scheduling problem (FJSP) is a generalised version of the classical job shop scheduling problem (JSP) and has been proven to be *NP*-hard (Pinedo 1995; Deng et al. 2017; Gao et al. 2019). It can be used to model a variety of real-world production scheduling cases. Theoretically, the FJSP consists of two sub-problems: (1) sequencing operations; and (2) assigning machines to the corresponding operations. In practice, however, there should be three sub-problems: (1) sequencing the operations; (2) assigning machines to the corresponding operations; and (3) assigning workers to the corresponding machines. The existing literature on flexible job shop scheduling has almost completely ignored human factors that are inextricably linked to production scheduling (Jensen 2002; Stephen et al. 2003; Neumann and Village 2012). These human factors, which include one's cognitive ability, aptitude, job knowledge, and job performance, may affect production costs and efficiency profoundly.

While only a limited number of studies considering human factors can be found in the literature of flexible job shop scheduling (e.g. see Zheng and Wang 2016; Gong et al. 2018), studies in other areas, such as advanced manufacturing systems (Udo and Ebiefung 1999), planning of production systems (Jensen 2002), medical service systems (Jensen 2002), dynamic production systems (Aryanezhad, Deljoo, and Mirzapour Al-e-Hashem 2009), and road maintenance systems (Brendan et al. 2011), have found many benefits derived from integrating human factors into manufacturing systems. These benefits include increased productivity, improved product quality, reduced production costs, decreased production throughput time, more reasonable worker arrangements, and so on.

Inspired by these findings, this paper presents a new model of the FJSP considering worker flexibility (FJSPW) to bridge the theoretical gap. We propose a hybrid artificial bee colony algorithm (HABCA) to solve the FJSPW, motivated by the fact that different variants of the artificial bee colony algorithm (ABCA) have been used to tackle production scheduling problems with great success (e.g. see Chakaravarthy et al. 2014; Gao, Suganthan, Chua, et al. 2015; Gao et al. 2016a; Sundar et al. 2017; Gong, Han, and Sun 2018). Unlike other ABCAs, our proposed HABCA is incorporated with problem-specific encoding and decoding as well as crossover and mutation operators. In addition, an effective local search method is designed to improve the speed of the algorithm and fully exploit the solution space.

As the formulated FJSPW is new, no existing benchmark instances from the literature are readily available for evaluation purposes. We therefore have to construct 13 FJSPW benchmark instances ourselves to test the HABCA's performance. These

*Corresponding author. Email: deng_arbeit@hnu.edu.cn

benchmark instances can later be used by other researchers to test their algorithms for the FJSPW. Extensive experiments carried out confirm that the performance of the HABCA is better compared to a hybrid genetic algorithm (GA) (Gao, Gen, and Sun 2006) and the ABCA from Wang et al. (2012).

The rest of this paper is organised as follows. In Section 2, we review human factor related research in different manufacturing systems. In Section 3, the mathematical model of the FJSPW is presented. In Section 4, details of the HABCA are described. Experiments and results are discussed in Section 5. Finally, we draw conclusion and outline future research directions in Section 6.

2. Background and related work

Researchers have long acknowledged the influence of human factors on production systems. From the hundreds of studies reviewed by Hunter (1986), it is clear that humans' cognitive ability is able to predict job performance. Felan and Fry (2001) showed that it is more beneficial to have a mixture of workers with no flexibility and workers with high flexibility over a full set of equally trained ones. Bidanda et al. (2005) studied the mutual influence of human factors and production systems, and found that workers' skill, training, communication, assignment, compensation/reward, team cooperation, autonomy and conflict handling are important human issues in manufacturing systems. Undoubtedly, some manufacturing sectors, especially those involving factory sites and team workers, cannot do without considering human factors, because human factors would lead to uncertainty in production time (Attia, Duquenne, and Le-Lann 2014; Gong et al. 2018; Stadnicka, Litwin, and Antonelli 2019). To keep production time as well as costs under control, it is vital to take human factors into consideration when studying production problems.

To provide a reference point for production managers to deal with human resource management, Wirojanagud et al. (2007) proposed an effective algorithm with the objective of minimising human costs. Aryanezhad, Deljoo, and Mirzapour Al-e-Hashem (2009) presented a method to handle production dynamics and worker arrangements simultaneously, and they pointed out that it is important to incorporate worker factors into a dynamic production system. Considering human fatigue and human recovery in production systems, a mixed-integer linear programming model was constructed by Jaber and Neumann (2010). Workers' learning curves on different manufacturing systems were studied by Corominas, Olivella, and Pastor (2010) and Jaber, Givi, and Neumann (2013). Attia, Duquenne, and Le-Lann (2014) found that worker productivity can be promoted by assigning flexible work and allocating suitable workers.

On flexible job shop scheduling related research, given its *NP*-hard nature, heuristic and metaheuristic algorithms are commonly used. Gao, Suganthan, Pan, et al. (2015) studied an FJSP with fuzzy processing time and developed a discrete harmony search algorithm for it. Lu et al. (2017) investigated the FJSP with controllable processing time and designed a discrete virus optimisation algorithm to solve it. Shahgholi Zadeh, Katebi, and Doniavi (2019) proposed a heuristic model for the dynamic FJSP considering variable processing time. Bożek and Werner (2018) presented a two-stage optimisation procedure for the FJSP with lot streaming and lot sizing of variable sublots, by which the makespan is minimised in the first stage, and the sum of sublot sizes of all operations and the number of operations that do not need to be split at all are maximised in the second stage. Ozturk, Bahadir, and Teymourifar (2018) proposed two new approaches for extracting composite priority rules for a dynamic multi-objective FJSP with the objectives of minimising the makespan, mean lateness, and mean flow time. Zhou, Yang, and Huang (2019) proposed three hyper-heuristic methods for co-evolution of machine assignment rules and job sequencing rules to solve the dynamic FJSP. Jun, Lee, and Chun (2019) presented a random forest-based approach for the FJSP with release time to minimise weighted tardiness, in which some dispatching rules are used to improve the performance of the algorithm.

The ABCA, which is our focus in this work, has been applied to different FJSPs with promising performance too. Wang et al. (2012) proposed an effective ABCA for the FJSP, hybridising it with a local search strategy and an updating mechanism. Wang et al. (2013) also introduced a hybridised ABCA, but for the fuzzy FJSP, with several strategies to generate good initial solutions and a local search based on variable neighbourhood search to enhance local intensification. Li, Pan, and Tasgetiren (2014) presented a novel discrete ABCA to solve a multi-objective FJSP with maintenance activities, aiming to minimise the makespan, total workload of machines, and workload of the critical machine. Gao et al. (2016a) presented an improved ABCA with a simple and effective heuristic rule to solve the FJSP with fuzzy processing time. Gao et al. (2016b) further addressed this FJSP with fuzzy processing time by considering new job insertion, and they designed a two-sage ABCA to solve it. Li et al. (2017) proposed a hybrid ABCA based on Tabu search with a rescheduling strategy to solve the FJSP by minimising the makespan. In one of the latest studies, Meng, Pan, and Sang (2018) presented a hybridised ABCA for an FJSP with overlapped operations aiming to minimise the total flow time.

Despite the success of ABCA variants in tackling FJSPs, it has not been used for the FJSPW. Cao and Yang (2011) presented an immune GA for an FJSP considering limited workers. Zheng and Wang (2016) studied a dual-resource constrained FJSP, taking operations' sequence as well as machine and worker arrangements into consideration. They solved

their problem using a knowledge-guided fruit fly optimisation algorithm. Gong et al. (2017) formulated a multi-objective FJSP in which the workers are flexible, and designed a memetic algorithm to deal with this problem. More recently, a double FJSP in which both machines and workers are flexible was proposed by Gong et al. (2018), and an effective hybrid evolutionary algorithm was used to solve it. It is worth pointing out that none of these studies had imposed any constraints on workers.

From the brief literature review above, we can see that (1) human factors should be taken into account in real-world production systems to improve production performance; (2) there is sufficient work considering human factors in manufacturing systems, which can be regarded as the basis and inspiration to study the FJSP with human factors; (3) existing research on the FJSPW has assumed that there are no constraints among workers, but in a real production environment a worker's production efficiency often greatly affects other workers and even the entire operation process (i.e. constraints among workers exist and should be considered). Therefore, in this work, we study the FJSPW in which workers' constraints are considered, and propose a HABCA to solve the problem.

3. Problem formulation

The proposed FJSPW is defined as follows: There are a set of n jobs, a set of m machines and a set of l workers; each job has a sequence of r operations to be processed one after another according to the precedence constraint; each operation must be processed by one worker selected from the worker set on one machine selected from the machine set. The processing time of each operation, which is operated by a worker on a machine, is fixed.

Taking Table 1 as an example, we see that there are two jobs, five machines and five workers. Each of the jobs has two operations. The machine sets to process O_{11}, O_{12}, O_{21} and O_{22} are [M_1, M_3 and M_5], [M_1 and M_2], [M_1, M_3 and M_4] and [M_4 and M_5], respectively; the worker sets to operate M_1, M_2, M_3, M_4 and M_5 are [W_1 and W_2], [W_2, W_4 and W_5], [W_1, W_2 and W_3], [W_4 and W_5] and [W_3 and W_4], respectively. The processing time is given in the 'Time' column.

In this work, the following assumptions have been made:

- At any time, each machine can process only one operation chosen from the corresponding operation set.
- At any time, each worker can operate only one machine chosen from the corresponding machine set.
- Each operation can be processed only once by a machine chosen from the corresponding machine set and operation constraints of all jobs should be satisfied.
- Each operation can be operated only once by a worker chosen from the corresponding worker set and operation constraints of all jobs should be satisfied.

Table 1. An example of a 2 × 5 × 5 FJSPW.

Jobs	Operations	Machines	Workers	Time
Job1	O_{11}	M_1	W_1	10
			W_2	15
		M_3	W_1	20
			W_2	10
			W_3	20
		M_5	W_3	15
			W_4	20
	O_{12}	M_1	W_1	15
			W_2	20
		M_2	W_2	15
			W_4	15
			W_5	10
Job2	O_{21}	M_1	W_1	10
			W_2	5
		M_3	W_1	20
			W_2	10
			W_3	20
		M_4	W_4	15
			W_5	20
	O_{22}	M_4	W_4	5
			W_5	10
		M_5	W_3	10
			W_4	15

- There are no precedence constraints among the operations of different jobs.
- It is not permitted to temporarily interrupt an operation after the operation has started.
- An operation of any job cannot be processed until its preceding operations are completed.
- Any operation of a job can be processed only if its preceding operations have been processed.
- The processing time is known in advance.

Indices

- i, h: Index of jobs, $i, h = 1,2, \ldots, n$;
- j, g: Index of operations, $j, g = 1,2, \ldots, r_i$;
- k: Index of machines, $k = 1,2, \ldots, m$;
- s: Index of workers, $k = 1,2, \ldots, l$;

Parameters

- n: Total number of jobs;
- r_i: Total number of operations of job i;
- m: Total number of machines;
- l: Total number of workers;
- O_{ij}: The *jth* operation of job i;
- t_{ijks}: Processing time of O_{ij} by worker s on machine k;
- CO_{ij}: Completion time of operation O_{ij};
- C_i: The completion time of job i;

Decision variables

- $X_{ijks} = \begin{cases} 1, & \text{if worker } s \text{ is selected to operate machine } k \text{ for operation } O_{ij} \\ 0, & \text{otherwise} \end{cases}$

The mathematical model is given as follows:

The objective function:
Minimising the makespan (f_1):

$$\min f_1 = \max_{1 \le i \le n} (C_i) \tag{1}$$

Linearisation of constraints

$$CO_{ij} - CO_{i(j-1)} \ge t_{ijks} X_{ijks} \quad \forall i = 1, \ldots, n; j = 2, \ldots, r_i; k = 1, \ldots, m; s = 1, \ldots, l \tag{2}$$

$$(CO_{hg} - CO_{ij} - t_{hgks}) X_{ijks} X_{hgks} \ge 0 \vee (CO_{ij} - CO_{hg} - t_{ijks}) X_{ijks} X_{hgks} \ge 0$$
$$\text{where, } \forall i = 1, \ldots, n; j = 1, \ldots, r_i; k = 1, \ldots, m; s = 1, \ldots, l \tag{3}$$

$$\sum_{j=1}^{r_i} X_{ijks} = 1 \quad \forall i = 1, \ldots, n; k = 1, \ldots, m; s = 1, \ldots, l \tag{4}$$

$$\sum_{k=1}^{m} X_{ijks} = 1 \quad \forall i = 1, \ldots, n; j = 1, \ldots, r_i; s = 1, \ldots, l \tag{5}$$

$$\sum_{s=1}^{l} X_{ijks} = 1 \quad \forall i = 1, \ldots, n; j = 1, \ldots, r_i; k = 1, \ldots, m \tag{6}$$

Here, Equation (1) is the objective of makespan minimisation. Equation (2) ensures the operation precedence constraint. Inequality (3) guarantees that each machine can handle only one operation at any time. Equation (4) guarantees that an operation chosen from the operation set can only be handled once. Equation (5) guarantees that only one machine is chosen from a corresponding machine set to process each operation. Equation (6) ensures that only one worker is assigned to each operation.

4. The proposed HABCA

4.1. The framework of the proposed HABCA

The ABCA is a metaheuristic approach that imitates the foraging behaviour of honeybee swarm (Li, Chiong, and Lin 2015; Karaboga and Akay 2009; Wang et al. 2012). Based on the ABCA, a HABCA has been proposed for the FJSPW in this work. There are four phases in the HABCA, which include the initialisation phase, employed bee phase, onlooker bee phase, and scout bee phase. In the initialisation phase, a new three-layer chromosome encoding method is used to explore the solution space more precisely. In the employed bee phase, μ employed bees randomly search for positions in a food area and assess the nectar density (which is the objective value for the algorithm) based on three vectors: operation sequence (OS), machine assignment (MA) and worker assignment (WA). A job-based crossover (JBX) operator is proposed for the OS vector and a random probability crossover (RPX) operator is proposed for the MA and WA vectors. In addition, three mutation methods are proposed for each of the vectors.

In the onlooker bee phase, an effective local search algorithm is developed, of which the onlooker bees 'fly' to where the food is according to information brought by the employed bees and search around the food source. Generally, a better food source would attract more onlooker bees, so a selection mechanism should be put in place. In this work, which food source should be chosen by an onlooker bee is determined by the following equation:

$$p_i = \frac{f(s_i)}{\sum_{j=1}^{\mu} f(s_j)}, \tag{7}$$

where $f(s_i)$ is the density of food source i and p_i denotes the probability of food source i to be chosen.

The scout bees are those that are currently searching for new food sources in the vicinity of the hive. In the scout bee phase, in order to enlarge the search space, a certain percentage of employed bees are eliminated. An identical number of scout bees are then generated to form a new population. The best bee is selected as the chosen solution in this population.

The overall framework of our HABCA can be found in Figure 1.

4.2. Chromosome encoding

A three-layer chromosome encoding method, which includes OS information, MS information, and WS information, is designed. Based on the data from Table 1, an example of how this encoding method works is shown in Figure 2(a), where the 1th–3rd rows represent the OS, MS, and WS information, respectively. In the first row, all operations are numbered as a series of consecutive integers. In this example, the operations of job 1 are numbered as 1 and 2, and the operations of job 2 are numbered as 3 and 4. Hence, the first row ([1, 3, 2, 4]) represents the operations [(O_{11}, O_{21}, O_{12}, O_{22})]. In the second row, each machine selected to process the corresponding operation is numbered as an integer according to its sequence in a predefined machine set. In this example, O_{11}, O_{12}, O_{21} and O_{22} are processed by machines M_3, M_2, M_1 and M_4, respectively. In the third row, each worker selected to process the corresponding operation is numbered as an integer according to its sequence in a predefined worker set. In this example, O_{11}, O_{12}, O_{21} and O_{22} are processed by workers W_2, W_5, W_2 and W_4, respectively.

4.3. Chromosome decoding

Both worker and machine constraints should be satisfied when decoding a chromosome for the proposed FJSPW. For example, the earliest start time of O_{21} in Figure 2(b) is 10 because its processing worker (W_2) is occupied on O_{11} from 0 to 10. We propose a new active decoding method inspired by Kacem, Hammadi, and Borne (2002), which is a well-known and effective activity decoding method for scheduling problems, ensuring that all operations can be processed as early as possible. Details of this decoding method are as follows:

Step 1: Obtain operation O_{ih} by converting a gene taken from the OS vector from left to right.

Step 2: Refer to the MA vector to get machine M_b to process O_{ih} and refer to the WA vector to get worker W_s to operate M_b; then refer to the timetable to obtain processing time P_{jhbs} of O_{ih};

Step 3: Find all the idle periods [ST_{mb}, ET_{mb}] of M_b, where ST_{mb} and ET_{mb} are the start time and end time, respectively; Find all the idle periods [ST_{ws}, ET_{ws}] of W_s, where ST_{ws} and ET_{ws} are the start time and end time, respectively. Equation (8) below can be used to calculate the earliest processing time t_a of O_{ih}:

$$t_a = \max\{C_{i(h-1)}, ST_{mb}, ST_{ws}\} \tag{8}$$

where $C_{i(h-1)} = 0$ if h equals 1.

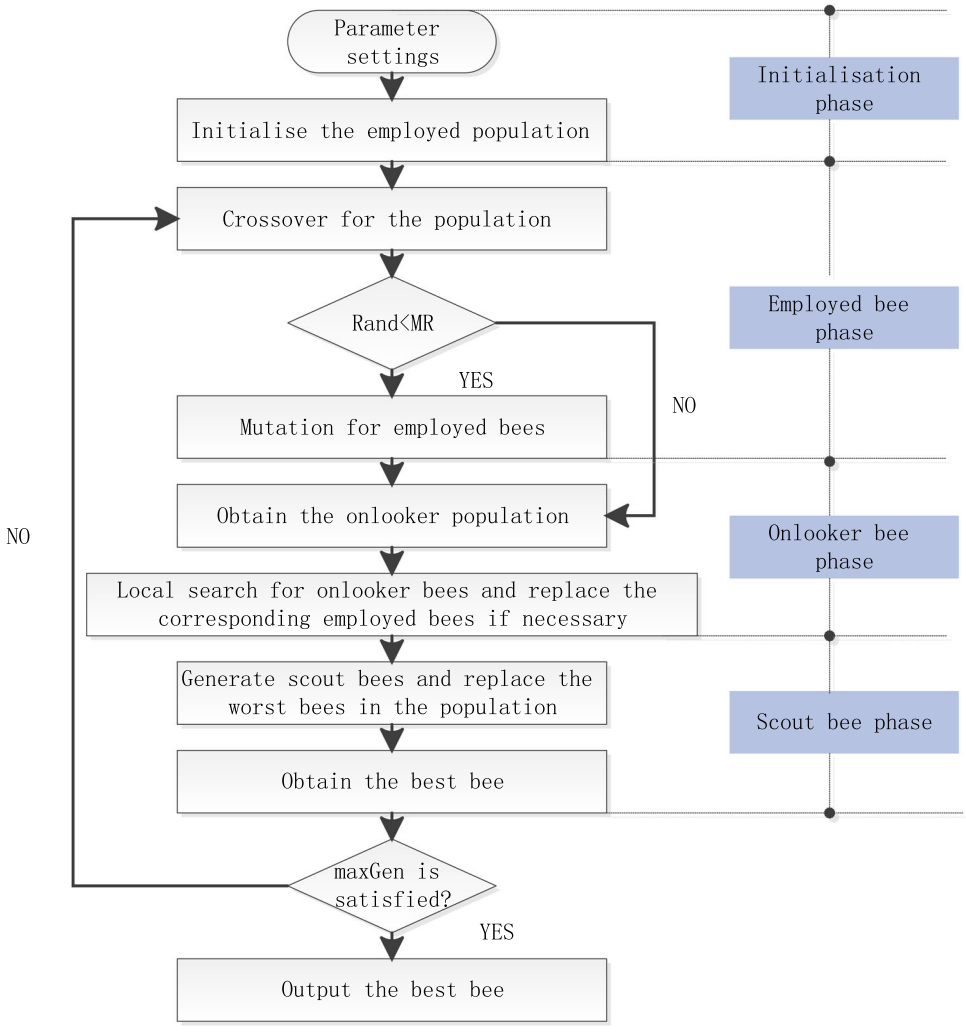

Figure 1. A framework of the proposed HABCA.

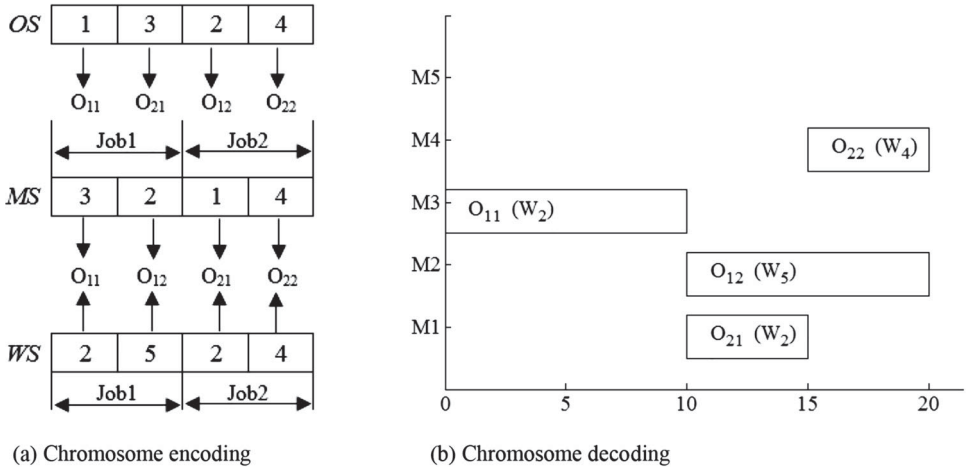

Figure 2. Encoding and decoding of a chromosome of the 2 × 5 × 5 FJSPW.

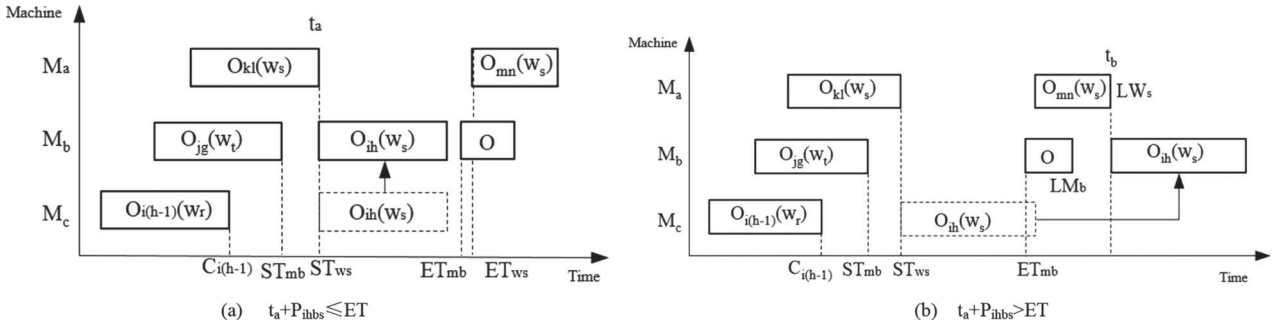

Figure 3. An active decoding method for the FJSPW.

Step 4: Use Equation (9) to determine whether O_{ih} can be inserted into a certain idle period:

$$t_a + P_{jhbs} \leq ET \tag{9}$$

where $ET = \min(ET_{mb}, ET_{ws})$.

If the idle period meets the insertion condition, O_{ih} can be inserted to this idle period; else, O_{ih} should be moved to the end of t_b as per the following equation:

$$t_b = \max(LM_b, LW_s) \tag{10}$$

where LM_b and LW_s are the end time of the last operation on machine b and worker s, respectively.

Figures 3 (a) and (b) illustrate examples of operation O_{ih} meeting and not meeting the condition of Equation (9), respectively.

Step 5: If all genes in the OS vector are scheduled, end the decoding method; else, return to Step 1.

4.4. Crossover operators

A JBX operator is proposed for OS and an RPX operator is proposed for MA and WA. Algorithm 1 shows the overall procedure. Details of JBX and RPX can be found in Algorithms 2 and 3, respectively. PA1 and PA2 are used to denote two parents, while CO1 and CO2 are the offspring.

Algorithm 1: Overall_Crossover (*PA* (*L*))

1: *PA'*$_1$ (*L*) ← *PA* (*L*)
2: **For** $j = 1$ to *GEN* **Do**
3: Select two parental chromosomes *PA1* and *PA2* randomly from *PA'*$_1$ (*L*)
4: [*CO1'*;*CO2'*] ← OS_Crossover (*PA1*, *PA2*)
5: [*CO1*;*CO2*] ← MAandWA_Crossover (*CO1'*, *CO2'*)
6: Copy the offspring chromosomes *CO1* and *CO2* in *PA'*$_1$ (*L*) to replace *PA1* and *PA2*, respectively
7: **End For**

Algorithm 2: OS_ Crossover (*PA1*, *PA2*)

1: obtain the OS vector [*PA1*; *PA2*]
2: **For** $i = 1$ to l **Do**
3: **If** random_probability $< CR$ **Then**
4: $OS\_Set = [OS\_Set, i]$
5: **End If**
6: **End For**
7: append the operations belonging to *OS_Set* in *PA1* to the same positions in *CO1*
8: copy the operations not belonging to *OS_Set* in *PA2* to the remaining empty positions in *CO1* from left to right
9: append the operations belonging to *OS_Set* in *PA2* to the same positions *of CO2*
10: copy the operations not belonging to *OS_Set* in *PA1* to the remaining empty positions in *CO2* from left to right
11: **Return** [*CO1*, *CO2*]

Algorithm 3: MAandWA _ Crossover (*PA1*, *PA2*)

obtain the MA & WA vectors [*PA1*; *PA2*]
randomly generate a binary vector unit (NN_{op})
set $i = 1$;
For $i = 1$ to NN_{op} **Do**
  If unit(i) $= 1$ Then
    $CO1(i) = PA1(i)$
    $CO2(i) = PA2(i)$
  Else
    $CO1(i) = PA2(i)$
    $CO2(i) = PA1(i)$
  End If
End For
Return [*CO1*, *CO2*]

4.5. Mutation operators

Three different operators are designed to mutate the OS, MA and WA vectors. Their overall procedure is shown in Algorithm 4. Details of these mutation operators for OS, MA and WA can be found in Algorithms 5–7, respectively.

Algorithm 4: Overall_Mutation (*PA* (*N*))

$PA'_2(N) \leftarrow \phi$
$PA'_{21}(N)$ = ProcessMutation($PA(N)$)
$PA'_{22}(N)$ = MachineMutaion($PA(N)$)
$PA'_{23}(N)$ = WorkerMutation($PA(N)$)
$PA'_2(N) = [PA'_{21}(N); PA'_{22}(N); PA'_{23}(N)]$;
While size($PA'_2(N)$) < *GEN*
  $PA'_2(N) = [PA'_2(N)$; newChrom]
End While

Algorithm 5: OS_Mutation (*PA (N)*)

$PA'_{21}(N) \leftarrow \phi$
For $j = 1$ to *GEN* **Do**
  If random_probability < *MR* **Then**
    select a gene (*og*) in the *OS* vector of individual (bee) *j* randomly
    ensure the allowable moving range ([start_og, end_og]) of element *og*
    select a position named *Posi* between [start_og, end_og] randomly
    move *og* to *Posi* to generate an *offspring*
    $PA'_{21}(N) = [PA'_{21}(N)$; *offspring*]
  End If
End For
Return $PA'_{21}(N)$

Algorithm 6: MA_Mutation (*PA* (*N*))

$PA'_{22}(N) \leftarrow \phi$
For $j = 1$ to *GEN* **Do**
  If random_probability < *MR* **Then**
select a gene (*og*) in the *OS* of individual (bee) *j* randomly
find the machine (*M*) to process *og* in MA
replace *M* with another machine selected from its available machine set to generate an *Offspring*
select a worker from the available worker set randomly to operate the *M*
$PA'_{22}(N) = [PA'_{22}(N)$; *Offspring*]
  End If
End For
Return $PA'_{22}(N)$

Algorithm 7: WA_Mutation (*PA (N)*)

1: $PA'_{23}(N) \leftarrow \phi$
2: **For** $j = 1$ to *GEN* **Do**
3: **If** random_probability $< MR$ **Then**
4: select a gene (og) in the *OS* of individual (bee) j randomly
5: find both machine (M) and worker (W) to process og randomly
6: replacing W with another worker selected from the available worker set to generate an *Offspring*
7: $PA'_{23}(N) = [PA'_{23}(N); Offspring]$
8: **End If**
9: **End For**
10: **Return** $PA'_{23}(N)$

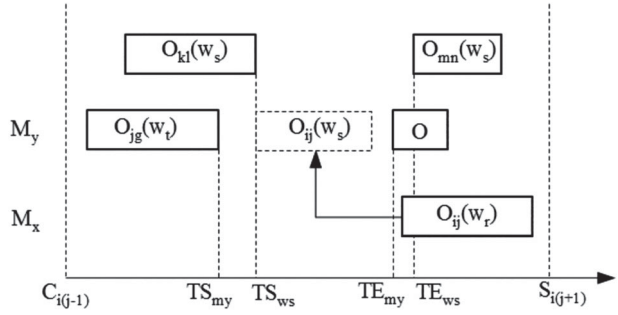

Figure 4. Local search for operations on the critical path.

4.6. The proposed local search

As shown in several previous studies (Yuan and Xu 2015; Wang et al. 2012; Li and Pan 2012), the only way to minimise the makespan of a schedule is to move its critical operations. Hence, we propose a local search operator based on critical operations of the FJSPW as follows:

Let PJ_{ij} and SJ_{ij} be the operations processed before and after O_{ij} of job i, respectively. Let PM^k_{ij} and SM^k_{ij} be the operations before and after $O_{i,j}$ on machine k, respectively. Let PW^s_{ij} and SW^s_{ij} be the operations before and after O_{ij} among the operations processed by worker s, respectively. As per Wang et al. (2012), the earliest start time of a critical operation, ST^E_{ij}, which equals $\max(CT^E(PJ_{ij}), CT^E(PM^k_{ij}), CT^E(PW^s_{ij}))$, where CT^E is the earliest completing time, should also be equal to its latest start time $ST^L_{i,j}$, satisfying $ST^L_{i,j} = CT^L_{ij} - P_{ij}$, and CT^L_{ij} is the latest completing time, equal to $\min(ST^L(SJ_{ij}), ST^L(SM^k_{ij}), ST^L(SW^s_{ij}))$. Generally speaking, several critical paths can be found by traversing a Gantt chart (like the one in Figure 4) from right to left. To obtain a unique critical path for conducting the local search operator, we choose the critical operations in this order: if there is a critical operation belonging to the same job of the current operation, this critical operation is selected as the next critical operation; else if there is a critical operation that will be processed by the same machine of the current operation, this critical operation is selected as the next critical operation; else if there is a critical operation that will be processed by the same worker of the current operation, this critical operation is selected as the next critical operation.

While carrying out the local search for O_{ij}, the machine and worker chosen from the available set are also changed accordingly. Suppose O_{ij} is assigned to machine y after O_{jg} and processed by worker s after O_{kl}, the earliest start time of O_{ij} can be obtained by the following equation:

$$ST^E_{ij} = \max(CT^E(PJ_{ij}), CT^E(PM^y_{ij}), CT^E(PW^s_{ij})) \tag{11}$$

If inequality (12) below is met, O_{ij} can be assigned to that position:

$$ST^E_{ij} + p_{ijys} \leq \min(ST^L(SJ_{ij}), ST^L(SM^y_{ij}), ST^L(SW^s_{ij})) \tag{12}$$

Figure 4 shows the local search method for operations on the critical path. As the local search operator finds better solutions by moving some operations one by one based on the critical paths, its computational complexity is $O(NM)$, where N is the total number of solutions and M is the total number of operations along the critical paths.

5. Experiments and results

The proposed HABCA was coded in MATLAB R2016a and implemented on a computer configured with an Intel Core i5 CPU of 3.3 GHz and 8GB RAM. Its performance was evaluated by comparing the results obtained with a hybrid GA by Gao, Gen, and Sun (2006) and an improved ABCA by Wang et al. (2012). All experiments, except those related to parameter tuning, were repeated 30 times given the stochastic nature of the algorithms being compared.

5.1. Constructing the FJSPW benchmarks

To verify the performance of the HABCA, we constructed 13 FJSPW benchmark instances – named FJSPWs 01–13 – based on the datasets from Brandimarte (1993) and Kacem, Hammadi, and Borne (2002), which are two of the most popular benchmark datasets for the FJSP that have been used by a large number of researchers. FJSPWs 01–03 are of smaller scale, generated based on the FJSP benchmark instances of Kacem, Hammadi, and Borne (2002); while FJSPWs 04–13 are large-scale instances generated based on those of Brandimarte (1993); both generated using Algorithm 8 detailed below. These benchmark instances can be downloaded from https://pan.baidu.com/s/1KTbqUvwmcBNgRp7CkWFQSA.

The FJSPW can be divided into total worker flexibility and partial worker flexibility. In this work, we considered partial worker flexibility, which means each machine can be operated by some of the workers.

Algorithm 8: GenerateNewBenchmark (*GNB*)

set all parameters for a normal distribution
obtain an operation, O_{ij}, and get the available machine set M_{ij} to process it
For $k = 1$: maximalSize (M_{ij})
  find the *kth* machine M_k to process O_{ij}; find the corresponding available worker set W_k to operate M_k; obtain the processing time P_{ijk} by referring to *GNB*
  For $s = 1$: maximalSize (W_k)
    generate P_{ijks}, which conforms to the normal distribution
    While $P_{ijks} \leq 0$
      regenerate the P_{ijks}
    End While
  End For
End For

5.2. Parameter settings

Our proposed HABCA has three critical parameters, which include the population size (PS), crossover rate (CR) and mutation rate (MR). We used the Taguchi method of Design of Experiments (DOE) (Montgomery 2008) based on FJSPWs 04 and 11 to obtain the best combination of these parameters. Specifically, we set each parameter to four levels, i.e. $PS = [50, 100, 150, 200]$, $CR = [0.2, 0.5, 0.7, 0.9]$, and $MR = [0.05, 0.1, 0.15, 0.2]$. For each possible configuration, the HABCA was run 10 times independently. Mean values of the best solutions were used to evaluate the outcome.

According to the DOE results, the trend of each factor level of FJSPW 04 and FJSPW 11 is illustrated in Figures 5 and 6, respectively. From Figures 5 and 6, we can see that PS is the most significant parameter. In Figure 5, we observe that when PS is set to 150 and 200, the makespan values for FJSPW 04 are 34.575 and 33.975, respectively. From Figure 6, we see that when PS is set to 150 and 200, the makespan values for FJSPW 11 are 522.475 and 517.200, respectively. This means that the makespan values of FJSPWs 04 and 11 are only reduced by 0.6 and 5.275, respectively, as the population sizes increase from 150 to 200. However, increasing the population size to 200 significantly increases the computation time. We therefore set the population size to 150 in our experiments.

In terms of CR, we see from Figure 5 that when it is set to 0.5 and 0.7, the makespan values for FJSPW 04 are 35.525 and 35.300, respectively. From Figure 6, when CR is set to 0.5 and 0.7, the makespan values for FJSPW 11 are 510.275 and 513.100, respectively. Here, the makespan values are reduced by 0.225 for FJSPW 04 and increased by 2.825 for FJSPW 11, when the crossover rate is changed from 0.5 to 0.7. Considering that a higher crossover probability can help the algorithm to fully explore the solution space and get out of local optima, we set the crossover rate to 0.7 in our experiments.

As for the mutation rate, we can clearly see from Figures 5 and 6 that the HABCA is able to obtain the best solution when it is set to 0.15. Therefore, we set the mutation rate to 0.15 in our experiments.

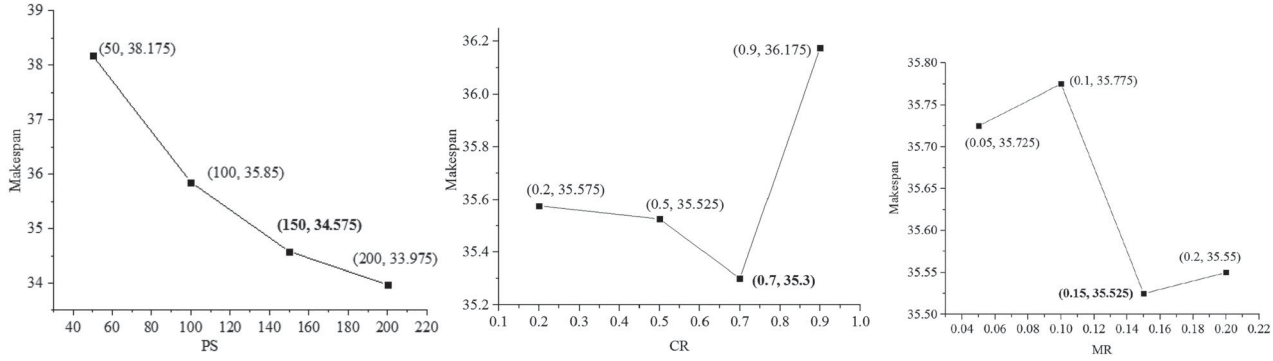

Figure 5. Factor level trends of FJSPW 04.

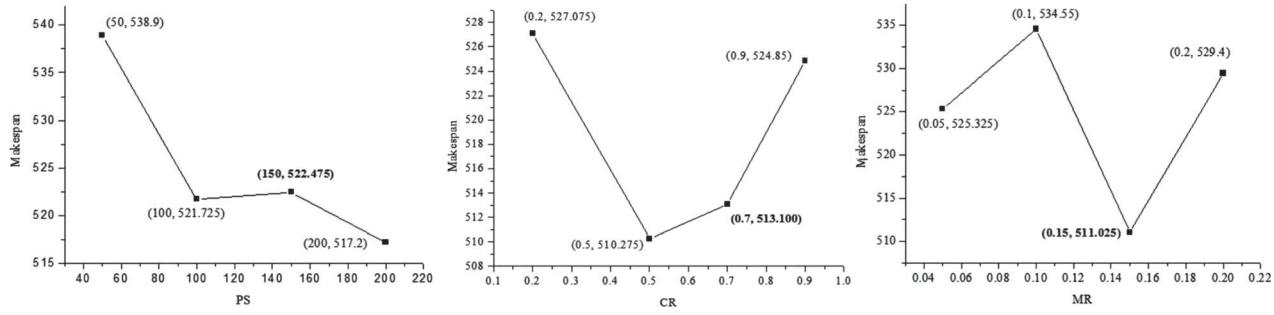

Figure 6. Factor level trends of FJSPW 11.

Table 2. Experimental results for FJSPWs 01–13 (those highlighted in bold are the best results among the three algorithms; and those marked with * indicate that they are significantly better than the results obtained by the other two algorithms).

		GA				ABCA				HABCA			
Benchmarks	J × M × W	Ma	Avg	SD	Time(s)	Ma	Avg	SD	Time(s)	Ma	Avg	SD	Time(s)
FJSPW 01	4 × 5 × 5	**7**	7.7	0.7	24.2	**7**	7.6	0.7	24.8	**7**	7.6	0.6	26.1
FJSPW 02	8 × 8 × 8	13	16.4	2.9	35.8	**11**	13.7	1.8	38.9	**11**	13.2	1.1	36.6
FJSPW 03	10 × 10 × 10	8	9.4	1.3	46.7	**7**	8.0	1.2	48.1	**7**	7.7*	0.7	47.3
FJSPW 04	10 × 6 × 6	**33**	36.8	1.4	118.3	**33**	35.9	1.8	113.5	**33**	35.2*	0.8	123.2
FJSPW 05	10 × 6 × 6	29	32.1	1.2	114.7	**24**	25.5	0.8	117.9	**24**	24.8*	1.1	122.3
FJSPW 06	15 × 8 × 8	203	223.1	12.6	148.9	198	199.8	1.3	146.7	**192**	194.3*	3.4	165.8
FJSPW 07	15 × 8 × 8	68	72.7	2.9	150.8	59	62.9	1.8	155.4	**55**	59.7*	3.2	160.1
FJSPW 08	15 × 4 × 4	192	198.8	7.1	135.2	185	196.1	4.8	137.6	**184**	192.3*	3.8	149.2
FJSPW 09	10 × 15 × 15	76	86.9	6.3	150.1	68	72.5	3.3	156.3	**65**	68.2*	2.4	164.7
FJSPW 10	20 × 5 × 5	167	180.4	6.6	214.7	**157**	166.5*	5.2	214.8	**157**	167.4	3.2	233.2
FJSPW 11	20 × 10 × 10	520	530.7	12.3	258.8	510	515.1	5.3	257.8	**505**	512.3*	4.5	266.6
FJSPW 12	20 × 10 × 10	397	420.8	18.9	273.2	380	397.3	8.6	277.3	**373**	392.4*	9.2	285.3
FJSPW 13	20 × 15 × 15	303	310.2	7.3	298.4	279	286.3	8.5	298.3	**266**	281.2*	9.3	310.4

5.3. Performance analysis

Based on the above parameter settings, we compared our proposed algorithm to a hybrid GA (Gao, Gen, and Sun 2006) and an improved ABCA (Wang et al. 2012) from the literature. The experimental results are shown in Table 2, in which J × M × W denotes the number of jobs, machines and workers (e.g. 4 × 5 × 5 means that this benchmark instance has 4 jobs, 5 machines and 5 workers), Ma is the best makespan among the 30 runs, Avg is the average value from the 30 runs, SD is the standard deviation of the 30 runs, and Time is the average computation time (in seconds). Best makespan values for each of the benchmark instances are marked in bold.

From Table 2, we can see that our HABCA shows competitive performance on the FJSPW. In terms of Ma, the HABCA is able to outperform the GA in 11 out of the 13 instances and shares the same results for the other 2 instances. The HABCA is also able to outperform the ABCA in 7 out of the 13 instances and has the same results for the other 6 instances. As for

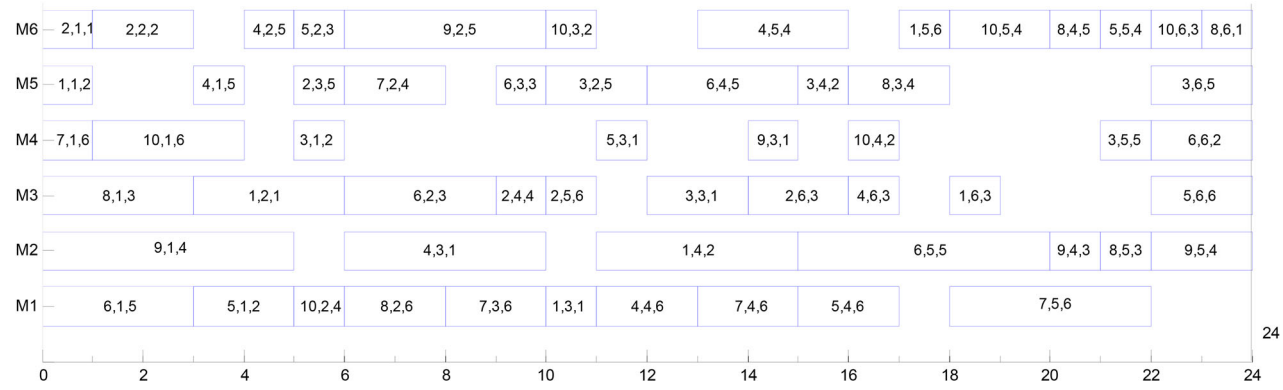

Figure 7. A Gantt chart illustrating the best solution for FJSPW 05.

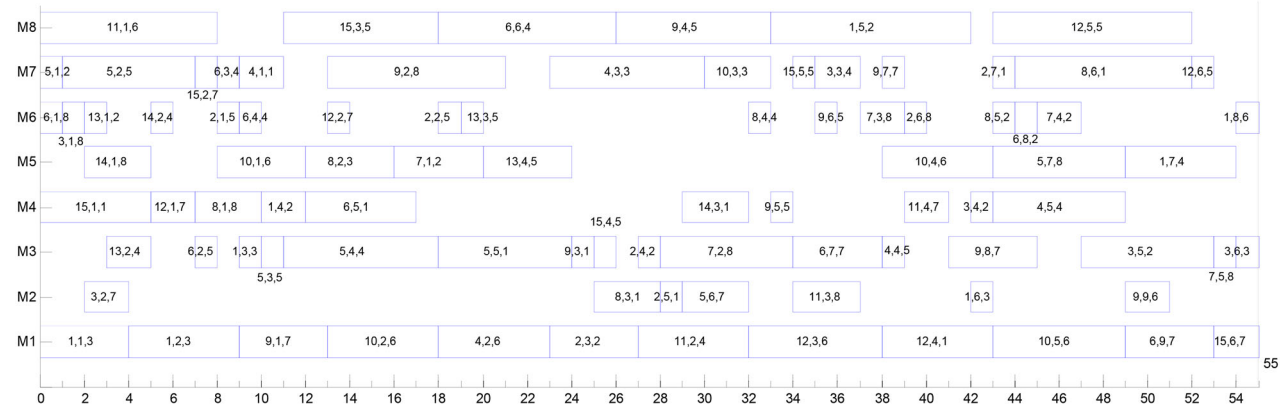

Figure 8. A Gantt chart illustrating the best solution for FJSPW 07.

the Avg, the HABCA outperforms both the GA and ABCA on all instances except FJSPW 01 and FJSPW 10 – on FJSPW 01, the HABCA has obtained almost identical results compared to the GA and ABCA; on FJSPW 10, the HABCA is better than the GA but not as good as the ABCA. For computation time, the HABCA performs slightly worse than both the GA and ABCA, which makes sense because it has a local search operator. Nevertheless, it is worth pointing out that, overall, the HABCA has managed to obtain the best solutions the three algorithms can possibly find in terms of Ma (highlighted in bold) for all the benchmark instances. In Figures 7 and 8, the Gantt charts of best solutions obtained by the HABCA on FJSPWs 05 and 07, respectively, are illustrated. In these figures, the numbers (x, y, z) indicate that the yth operation of job x is processed by worker z.

To check whether the observed differences in performance were significant, we applied the Mann–Whitney U test with a significance level of 0.05 to the results from 30 runs. Statistical analysis showed that our HABCA performs significantly better than both the GA and ABCA on most of the benchmark instances except FJSPWs 01, 02, and 10 (see the results marked with * in Table 2). FJSPWs 01 and 02 are small-scale instances, so it is not surprising that there are no significant differences between results obtained by the algorithms. For FJSPW 10, the ABCA performs significantly better than our HABCA. Nevertheless, we can clearly see the advantages of our proposed HABCA on large-scale instances such as FJSPWs 11–13.

To inspect the convergence speed of the proposed algorithm, we selected two small-scale and two large-scale instances, namely FJSPWs 02, 03, 06 and 12, and depicted the convergence curves of our HABCA as well as the GA and ABCA on these instances in Figures 9 (a)–(d). From the figures, we can clearly see that the proposed HABCA has faster convergence speed compared to the GA and ABCA.

Summing up, the proposed HABCA is able to obtain more better solutions than the GA and ABCA, and it also has faster convergence speed. With benchmark instances of larger scale, such as FJSPWs 11-13, the proposed HABCA is obviously superior compared to the GA and ABCA. We can therefore conclude that the proposed algorithm is more effective than the algorithms in comparison, especially in solving large-scale FJSPW instances. The strong results can be attributed to

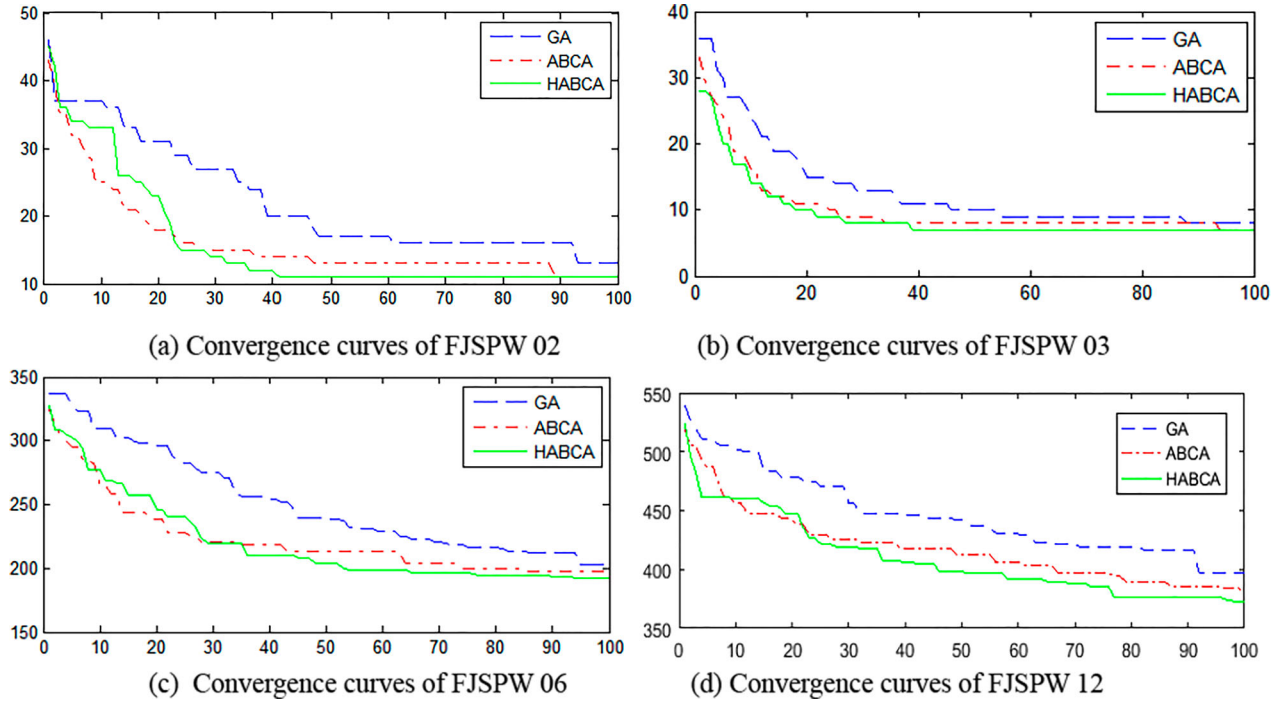

Figure 9. Convergence curves.

the useful chromosome encoding and decoding methods, as well as the crossover and mutation operators, and the speed-up strategy in the proposed local search.

6. Conclusion and future work

In this paper, we presented a new flexible job-shop scheduling model considering worker flexibility. This model – FJSPW – is an extension of the FJSP, which is commonly seen in real production environments. Our proposed model can help production managers in manufacturing companies to schedule their production activities considering machine flexibility and worker flexibility simultaneously. With this integrated model, imbalanced resource loads and unexpected bottlenecks, which regularly arise in traditional models that ignore worker flexibility, can be easily avoided.

We also proposed a hybrid algorithm, the HABCA, to solve the FJSPW. This hybrid algorithm has a well-designed three-layer chromosome encoding method, a new chromosome decoding method, and some effective crossover and mutation operators. An innovative local search approach has also been designed to improve the speed of the algorithm and fully exploit the solution space. A total of 13 benchmark instances were constructed based on some specific characteristics of the FJSPW to test the performance of the HABCA. Experimental results showed that the proposed HABCA performs exceptionally well in solving the FJSPW instances, especially on large-scale ones. These benchmark instances can later be used by other researchers to evaluate their algorithms against ours.

To conclude, our work here not only has practical significance in terms of the model proposed, but also research significance, as the algorithm proposed and benchmark instances created can provide a reference point for other researchers to design optimisation algorithms with local search operators and test their algorithms using our benchmark instances. For future work, we will study the FJSPW in more detail. For example, the redistribution of job operations and machine breakdowns can be considered and incorporated into the model.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the National Key R&D Program of China [grant number 2018YFB1701400]; the National Natural Science Foundation of China [grant number 71473077]; the State Key Laboratory of Advanced Design and Manufacturing for Vehicle Body, Hunan University [grant number 71775004]; and the State Key Laboratory of Construction Machinery [grant number SKLCM2019-03].

ORCID

Raymond Chiong http://orcid.org/0000-0002-8285-1903

References

Aryanezhad, M., V. Deljoo, and S. M. J. Mirzapour Al-e-Hashem. 2009. "Dynamic Cell Formation and the Worker Assignment Problem: A New Model." *The International Journal of Advanced Manufacturing Technology* 41 (3-4): 329–342.

Attia, E., P. Duquenne, and J. M. Le-Lann. 2014. "Considering Skills Evolutions in Multi-Skilled Workforce Allocation with Flexible Working Hours." *International Journal of Production Research* 52 (15): 4548–4573.

Bidanda, B., P. Ariyawongrat, K. L. Needy, B. A. Norman, and W. Tharmmaphornphilas. 2005. "Human Related Issues in Manufacturing Cell Design, Implementation, and Operation: A Review and Survey." *Computers & Industrial Engineering* 48 (3): 507–523.

Bożek, A., and F. Werner. 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.

Brendan, R., Q. Rong, S. Alex, and P. Tony. 2011. "Integrating Human Factors and Operational Research in a Multidisciplinary Investigation of Road Maintenance." *Ergonomics* 54 (5): 436–452.

Cao, X., and Z. Yang. 2011. "An Improved Genetic Algorithm for Dual-Resource Constrained Flexible Job Shop Scheduling." Paper presented at the Fourth International Conference on Intelligent Computation Technology and Automation.

Chakaravarthy, G. V., S. Marimuthu, S. G. Ponnambalam, and G. Kanagaraj. 2014. "Improved Sheep Flock Heredity Algorithm and Artificial Bee Colony Algorithm for Scheduling m-Machine Flow Shops lot Streaming with Equal Size Sub-Lot Problems." *International Journal of Production Research* 52 (5): 1509–1527.

Corominas, A., J. Olivella, and R. Pastor. 2010. "A Model for the Assignment of a Set of Tasks When Work Performance Depends on Experience of all Tasks Involved." *International Journal of Production Economics* 126 (2): 335–340.

Deng, Q., G. Gong, X. Gong, L. Zhang, W. Liu, and Q. Ren. 2017. "A Bee Evolutionary Guiding Nondominated Sorting Genetic Algorithm II for Multiobjective Flexible Job-Shop Scheduling." *Computational Intelligence and Neuroscience* 2017: 1–20.

Felan, J. T., and T. D. Fry. 2001. "Multi-Level Heterogeneous Worker Flexibility in a Dual Resource Constrained (DRC) Job-Shop." *International Journal of Production Research* 39 (14): 3041–3059.

Gao, J., M. Gen, and L. Y. Sun. 2006. "Scheduling Jobs and Maintenances in Flexible Job Shop with a Hybrid Genetic Algorithm." *Journal of Intelligent Manufacturing* 17 (4): 493–507.

Gao, K. Z., P. N. Suganthan, T. J. Chua, C. S. Chong, T. X. Cai, and Q. K. Pan. 2015. "A Two-Stage Artificial Bee Colony Algorithm Scheduling Flexible Job-Shop Scheduling Problem with New Job Insertion." *Expert Systems with Applications* 42 (21): 7652–7663.

Gao, K. Z., P. N. Suganthan, Q. K. Pan, T. J. Chua, C. S. Chong, and T. X. Cai. 2016a. "An Improved Artificial Bee Colony Algorithm for Flexible Job-Shop Scheduling Problem with Fuzzy Processing Time." *Expert Systems with Applications* 65: 52–67.

Gao, K. Z., P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren. 2015. "An Effective Discrete Harmony Search Algorithm for Flexible Job Shop Scheduling Problem with Fuzzy Processing Time." *International Journal of Production Research* 53 (19): 5896–5911.

Gao, K. Z., P. N. Suganthan, Q. K. Pan, M. F. Tasgetiren, and A. Sadollah. 2016b. "Artificial Bee Colony Algorithm for Scheduling and Rescheduling Fuzzy Flexible Job Shop Problem with New Job Insertion." *Knowledge-Based Systems* 109: 1–16.

Gao, K., F. Yang, M. Zhou, Q. Pan, and P. N. Suganthan. 2019. "Flexible Job-Shop Rescheduling for New Job Insertion by Using Discrete Jaya Algorithm." *IEEE Transactions on Cybernetics* 49 (5): 1944–1955.

Gong, X., Q. Deng, G. Gong, W. Liu, and Q. Ren. 2017. "A Memetic Algorithm for Multi-Objective Flexible Job-Shop Problem with Worker Flexibility." *International Journal of Production Research* 56 (7): 2506–2522.

Gong, G. L., Q. W. Deng, X. R. Gong, W. Liu, and Q. H. Ren. 2018. "A New Double Flexible job-Shop Scheduling Problem Integrating Processing Time, Green Production, and Human Factor Indicators." *Journal of Cleaner Production* 174: 560–576.

Gong, D. W., Y. Y. Han, and J. Y. Sun. 2018. "A Novel Hybrid Multi-Objective Artificial bee Colony Algorithm for Blocking Lot-Streaming Flow Shop Scheduling Problems." *Knowledge-Based Systems* 148: 115–130.

Hunter, J. E. 1986. "Cognitive Ability, Cognitive Aptitudes, Job Knowledge, and Job Performance." *Journal of Vocational Behavior* 29 (3): 340–362.

Jaber, M. Y., Z. S. Givi, and W. P. Neumann. 2013. "Incorporating Human Fatigue and Recovery into the Learning-Forgetting Process." *Applied Mathematical Modelling* 37 (12-13): 7287–7299.

Jaber, M. Y., and W. P. Neumann. 2010. "Modelling Worker Fatigue and Recovery in Dual-Resource Constrained Systems." *Computers & Industrial Engineering* 59 (1): 75–84.

Jensen, P. L. 2002. "Human Factors and Ergonomics in the Planning of Production." *International Journal of Industrial Ergonomics* 29 (3): 121–131.

Jun, S., S. Lee, and H. Chun. 2019. "Learning Dispatching Rules Using Random Forest in Flexible Job Shop Scheduling Problems." *International Journal of Production Research* 57 (10): 3290–3310.

Kacem, I., S. Hammadi, and P. Borne. 2002. "Pareto-Optimality Approach for Flexible Job-Shop Scheduling Problems: Hybridization of Evolutionary Algorithms and Fuzzy Logic." *Mathematics and Computers in Simulation* 60 (3-5): 245–276.

Karaboga, D., and B. Akay. 2009. "A Comparative Study of Artificial Bee Colony Algorithm." *Applied Mathematics and Computation* 214 (1): 108–132.

Li, B., R. Chiong, and M. Lin. 2015. "A Balance-Evolution Artificial bee Colony Algorithm for Protein Structure Optimization Based on a Three-Dimensional AB Off-Lattice Model." *Computational Biology & Chemistry* 54: 1–12.

Li, J. Q., and Q. K. Pan. 2012. "Chemical-Reaction Optimization for Flexible Job-Shop Scheduling Problems with Maintenance Activity." *Applied Soft Computing* 12 (9): 2896–2912.

Li, J. Q., Q. K. Pan, and M. F. Tasgetiren. 2014. "A Discrete Artificial Bee Colony Algorithm for the Multi-Objective Flexible Job-Shop Scheduling Problem with Maintenance Activities." *Applied Mathematical Modelling* 38 (3): 1111–1132.

Li, X., Z. Peng, B. Du, J. Guo, W. Xu, and K. Zhuang. 2017. "Hybrid Artificial Bee Colony Algorithm with a Rescheduling Strategy for Solving Flexible Job Shop Scheduling Problems." *Computers & Industrial Engineering* 113: 10–26.

Lu, C., X. Y. Li, L. Gao, W. Liao, and J. Yi. 2017. "An Effective Multi-Objective Discrete Virus Optimization Algorithm for Flexible Job-Shop Scheduling Problem with Controllable Processing Times." *Computers & Industrial Engineering* 104: 156–174.

Meng, T., Q. K. Pan, and H. Y. Sang. 2018. "A Hybrid Artificial Bee Colony Algorithm for a Flexible Job Shop Scheduling Problem with Overlapping in Operations." *International Journal of Production Research* 56 (16): 5278–5292.

Montgomery, D. C. 2008. *Design & Analysis of Experiments*. New York: John Wiley and Sons.

Neumann, W. P., and J. Village. 2012. "Ergonomics Action Research II: A Framework for Integrating HF into Work System Design." *Ergonomics* 55 (10): 1140–1156.

Ozturk, G., O. Bahadir, and A. Teymourifar. 2018. "Extracting Priority Rules for Dynamic Multi-Objective Flexible Job Shop Scheduling Problems Using Gene Expression Programming." *International Journal of Production Research* 57 (10): 3121–3137.

Pinedo, M. 1995. *Scheduling: Theory, Algorithms, and Systems*. Englewood Cliffs, NJ: Prentice Hall.

Shahgholi Zadeh, M., Y. Katebi, and A. Doniavi. 2019. "A Heuristic Model for Dynamic Flexible Job Shop Scheduling Problem Considering Variable Processing Times." *International Journal of Production Research* 57 (10): 3020–3035.

Stadnicka, D., P. Litwin, and D. Antonelli. 2019. "Human Factor in Intelligent Manufacturing Systems – Knowledge Acquisition and Motivation." *Procedia CIRP* 79: 718–723.

Stephen, B., O'Brien Pallas Linda, A. Chris, T. M. Gail, and T. Donna. 2003. "Beyond Demographic Change in Human Resources Planning: An Extended Framework and Application to Nursing." *Journal of Health Services Research & Policy* 8 (4): 225–229.

Sundar, S., P. N. Suganthan, C. T. Jin, C. T. Xiang, and C. C. Soon. 2017. "A Hybrid Artificial Bee Colony Algorithm for the Job-Shop Scheduling Problem with No-Wait Constraint." *Soft Computing* 21 (5): 1193–1202.

Udo, G. G., and A. A. Ebiefung. 1999. "Human Factors Affecting the Success of Advanced Manufacturing Systems." *Computers & Industrial Engineering* 37 (1–2): 297–300.

Wang, L., G. Zhou, Y. Xu, and M. Liu. 2013. "A Hybrid Artificial Bee Colony Algorithm for the Fuzzy Flexible job-Shop Scheduling Problem." *International Journal of Production Research* 51 (12): 3593–3608.

Wang, L., G. Zhou, Y. Xu, S. Wang, and M. Liu. 2012. "An Effective Artificial Bee Colony Algorithm for the Flexible job-Shop Scheduling Problem." *The International Journal of Advanced Manufacturing Technology* 60 (1-4): 303–315.

Wirojanagud, P., E. S. Gel, J. W. Fowler, and R. Cardy. 2007. "Modelling Inherent Worker Differences for Workforce Planning." *International Journal of Production Research* 45 (3): 525–553.

Yuan, Y., and H. Xu. 2015. "Multiobjective Flexible Job Shop Scheduling Using Memetic Algorithms." *IEEE Transactions on Automation Science and Engineering* 12 (1): 336–353.

Zheng, X., and L. Wang. 2016. "A Knowledge-Guided Fruit Fly Optimization Algorithm for Dual Resource Constrained Flexible Job-Shop Scheduling Problem." *International Journal of Production Research* 54 (18): 5554–5566.

Zhou, Y., J. Yang, and Z. Huang. 2019. "Automatic Design of Scheduling Policies for Dynamic Flexible Job Shop Scheduling via Surrogate-Assisted Cooperative Co-Evolution Genetic Programming." *International Journal of Production Research*. doi:10.1080/00207543.2019.1620362.