J Intell Manuf (2017) 28:1961–1972
DOI 10.1007/s10845-015-1082-0

A hybrid discrete particle swarm optimization for dual-resource constrained job shop scheduling with resource flexibility

Jing Zhang[1] · **Wanliang Wang**[2] · **Xinli Xu**[2]

Received: 13 August 2014 / Accepted: 16 April 2015 / Published online: 29 April 2015

Abstract In this paper, a novel hybrid discrete particle swarm optimization algorithm is proposed to solve the dual-resource constrained job shop scheduling problem with resource flexibility. Particles are represented based on a three-dimension chromosome coding scheme of operation sequence and resources allocation. Firstly, a mixed population initialization method is used for the particles. Then a discrete particle swarm optimization is designed as the global search process by taking the dual-resources feature into account. Moreover, an improved simulated annealing with variable neighborhoods structure is introduced to improve the local searching ability for the proposed algorithm. Finally, experimental results are given to show the effectiveness of the proposed algorithm.

Keywords Particle swarm optimization · Simulated annealing · Dual-resource constraint · Resource flexibility

Introduction

Job shop scheduling problem (JSSP) is a well-known combinational optimization problem which has found a lot of applications in real manufacturing industry. In classical JSSP, it is assumed that jobs are processed on pre-given machines and that the constraints of worker resources are neglected. Such assumptions reduced the problem complexity and fruitful research results were obtained, see (Qiu and Lau 2014; González et al. 2013; Adibi and Shahrabi 2014) for example and references therein. However, worker resources need taking into consideration since the labor cost is becoming much higher in manufactory fields nowadays. On the other hand, an operation can always be processed on various machines and with various workers, which means the resources are flexible (Nie et al. 2013; Pérez and Raupp 2014). The main advantage of resource flexibility is the increasing flexibility of the whole scheduling process, which is potential to save production time or cost. Thus a more general case which considers dual-resources constrained (DRC) JSSP with resource flexibility has been attracting increasing attention (Xu et al. 2011) and was presented in Gargeya and Deane (1996). In what follows, DRC JSSP with resource flexibility is called DRC-FJSSP for short.

The task of DRC-FJSSP is to determine both the operation sequence and the resources allocation by optimize certain objectives with constraints. Clearly, DRC-FJSSP is NP-hard and much more involved than classical JSSP. One commonly used way is first to obtain the operation sequence by intelligent algorithms such as Genetic algorithm (GA) (ElMaraghy et al. 1999, 2000) and ant colony algorithms (Li et al. 2011a, b), and then to allocate the machine and worker resources via dispatching rules. The dispatching rules include Earliest Finish Time, Shortest Process Time and so on. It is easy for these methods to find a local optimal solution and cost less computational time. However, the global optimal solution is difficult to find in this way. One effective way to overcome this drawback is to determine the operation sequence and resources allocation simultaneously via

✉ Jing Zhang
bay_229@163.com

Wanliang Wang
wangwanliang@zjut.edu.cn

1 Department of Computer and Information Technology, Zhejiang Police College, Hangzhou, People's Republic of China

2 College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou, People's Republic of China

hybrid intelligent algorithms. In Li et al. (2010), an improved inherited GA with ant colony algorithms was presented to solve DRC-FJSSP. A four-dimensional chromosome coding scheme was used, and the resource crossover and mutation operator were designed to improve the global searching ability. In Lei and Guo (2014), an improved variable neighborhood search was proposed to for the DRC-FJSSP. This method has the ability of exploring different neighborhoods of the current solution and thus is easy for implementation. However, the obtained solutions may contain infeasible ones in these methods.

Since searching mechanism is important in finding global solutions of optimization problems, particle swarm optimization (PSO) algorithm is considered in this paper due to its good feature on it. In PSO, particle can learn from the global best and the personal best and thus the global and local searching ability are balanced.

PSO was first proposed in Kennedy and Eberhart (1995) for continuous optimization problems and the main advantage is fewer control parameters in the continuous space (Katherasan et al. 2014; Coello et al. 2004). However, PSO is considerably limited to the combinatorial optimization problems because the updating process of particle position was carried out in continuous domain. Thus PSO need modifying to solve combinatorial optimization problems. For example, PSO had been applied to the flow-shop scheduling problem (Vijay Chakaravarthy et al. 2013; AitZai et al. 2014), the JSSP (Zhang and Wu 2010; Lin et al. 2010), and the resource constraint project scheduling (Zhang et al. 2006), but there are few results applying PSO to solve DRC-FJSSP. Hence solving DRC-FJSSP with Hybrid PSO is an interesting problem to be investigated and has not been studied to be the best of the authors' knowledge.

In this paper, a hybrid discrete PSO (HDPSO) is presented to solve the DRC-FJSSP with minimizing makespan being the objective function. A three-dimension chromosome coding scheme is used for the particles, which are the operation sequence, machine and worker allocation, respectively. The particles are initialized in a mixed way. Then a discrete PSO (DPSO) is designed as the global search process, in which the position updating equations are modified to avoid infeasible solutions. Moreover, an improved simulated annealing (ISA) with variable neighborhoods structure is introduced to improve the local searching ability for the proposed algorithm. Finally, experimental results show that the proposed algorithm is feasible and effective.

The remainder of this paper is organized as follows. "Problem description" section formally defines the scheduling model and presents a mathematical formulation. The HDPSO algorithm is presented in "HDPSO for DRC-FJSSP" section. In "Experiment results" section, experimental results are given and discussed. Conclusions are made together with future research direction in "Conclusions" section.

Problem description

Notations

p	Job index
q	Operation index
h	Machine index
g	Worker index
O_{pq}	qth operation of job p
J	Set of jobs
M	Set of machines
W	Set of workers
M_{pq}	Set of candidate machines for O_{pq}
W_h	Set of candidate workers for machine h

Parameters

n	Total number of jobs
m	Total number of machines
b	Total number of workers
n_p	Total operation number of job p
t_{pqh}	Processing time of operation O_{pq} on machine h
U	A large number

Decision variables

S_{pq}	Starting time of operation O_{pq}
K_{hg}	Starting time of machine h processed by worker g
C_{pq}	Completing time of operation O_{pq}
C_p	Completing time of job p
$C_{\max}$	Maximum completing time of all jobs

$$\sigma_{pqh} = \begin{cases} 1, & \text{if machine } h \text{ is selected for the operation } O_{pq} \\ 0, & \text{otherwise} \end{cases}$$

$$\varepsilon_{hg} = \begin{cases} 1, & \text{if worker } g \text{ is selected for the machine } h \\ 0, & \text{otherwise} \end{cases}$$

$$\chi_{hg-h'g} = \begin{cases} 1, & \text{if worker } g \text{ is performed on machine } h \text{ before } h' \\ 0, & \text{otherwise} \end{cases}$$

$$\varsigma_{pqh-p'q'h} = \begin{cases} 1, & \text{if } O_{pq} \text{ is performed on machine } h \text{ before } O_{p'q'} \\ 0, & \text{otherwise} \end{cases}$$

Problem formulation

Before formulating the DRC-FJSSP, assumptions A1–A4 are made as follows.

A1 All jobs, machines and workers are available at time zero.

A2 Each machine can process only one operation at one time.
A3 Every worker can operate only one machine at one time.
A4 Preemption is not allowed, i.e., any operation cannot be interrupted until its completion once it is started.

Under the assumptions and notations, the mathematical model for the problem is defined as follows:

$$F = \min C_{\max} = \min \left\{ \max_{p=1}^{n} \{C_p\} \right\}; \tag{1}$$

s.t.

$$S_{p(q+1)} \geq S_{pq} + t_{pqh}\sigma_{pqh}\varepsilon_{hg},\ p \in J;\ h \in M_{pq};\ g \in W_h;\quad q = 1, 2, \ldots, n_p - 1; \tag{2}$$

$$S_{p'q'} + (1 - \varsigma_{pqh-p'q'h})U \geq S_{pq} + t_{pqh},\ p,\ p' \in J;\ h \in M_{pq};\ q,\quad q' = 1, 2, \ldots, n_p; \tag{3}$$

$$K_{h'g} + (1 - \chi_{hg-h'g})U \geq K_{hg},\ h, h' \in M_{pq};\ g \in W_h; \tag{4}$$

$$K_{hg} + (1 - \sigma_{pqh})(1 - \varepsilon_{hg})U \leq S_{pq},\ p \in J;\ h \in M_{pq};\ g \in W_h;\quad q = 1, 2, \ldots, n_p; \tag{5}$$

$$S_{p(q+1)} + (1 - \varsigma_{p(q+1)h-p'q'h})U \geq C_{pq},\ p,\ p' \in J;\ h \in M_{pq};\ q,\quad q' = 1, 2, \ldots, n_p - 1; \tag{6}$$

$$\sum_{h=1}^{M_{pq}} \sigma_{pqh} = 1,\ p \in J;\ q = 1, 2, \ldots n_p; \tag{7}$$

$$\sum_{g=1}^{W_h} \varepsilon_{hg} = 1,\quad h = 1, 2, \ldots m; \tag{8}$$

$$C_p \leq C_{\max},\ p \in J; \tag{9}$$

$$S_{pq}, t_{pqh} \geq 0,\ p \in J;\quad q = 1, 2, \ldots n_p;\ h = 1, 2, \ldots, m; \tag{10}$$

$$\sigma_{pqh}, \varsigma_{pqh-p'q'h} \in \{0, 1\},\ p, p' \in J;\ h \in M_{pq};\ q, q' = 1, 2, \ldots, n_p; \tag{11}$$

$$\varepsilon_{hg}, \chi_{hg-h'g} \in \{0, 1\},\ h, h' \in M_{pq};\ g \in W_h; \tag{12}$$

where the objective function F is the minimization of the maximal makespan. Constraint (2) ensures that the precedence relationships between the operations of a job are not violated. Constraint (3) implies that each machine cannot process more than one operation at the same time. Constraint (4) implies that each worker cannot process more than one machine in the same time. Constraint (5) makes sure that each operation can be started after the worker selected for the processing machine is available. Constraint (6) makes sure that each operation can be started after the previous is completed. Constraint (7) makes sure that each operation is assigned to only one machine from its candidate machines set. Constraint (8) makes sure that each machine is assigned to only one worker from its candidate workers set. Constraint (9) implies the maximum completion time. Constraint (10) implies the start time and the processing time of each operation. Constraints (11) and (12) state the range of decision variables.

HDPSO for DRC-FJSSP

The HDPSO algorithm is presented in this section. A coding scheme and a mixed population initialization way are given first. Then a DPSO is presented for global search which updates particles in discrete domain directly. Finally, an ISA with variable neighborhood is introduced to improve the local search ability.

Coding scheme

The solution of DRC-FJSSP is a combination of operation sequences and resources allocation. Inspired by the chromosome coding scheme of GA (Lei 2010), we integrate the operation sequence vector and the resources allocation vectors as the particle coding scheme of HDPSO. Thus each particle has three vectors denoted by (OS, MA, WA), where OS represents the operation sequence, MA and WA represents the machine allocation and worker allocation, respectively. The length of each vector is $L = \sum_{p=1}^{n} n_p$.

For OS, the number p, $p = 1, 2, \ldots, n$, appears n_p times and the qth appearance of p means the operation O_{pq}. Thus an element in OS stands for an operation. An element in MA and WA refers to the index of the machine and the worker, respectively. Obviously, they are subject to constraints (7) and (8), respectively. The three vectors compose a matrix and each column of the matrix stands for job in which a worker makes an operation in a machine and the corresponding costing time is t_{pqh}. Take Tables 1 and 2 for example, in which $n = 3, m = 4, b = 3, n_1 = 4, n_2 = 3$ and $n_3 = 3$, then it follows that $L = 10$. Table 3 shows a code of a particle and the second column (2,1,1) means that worker 1 take operation O_{21} on machine 1 and the corresponding costing time is 15.

Population initialization

To guarantee an initial population with certain quality and diversity, some strategies are utilized in a mixed way. The initialization process includes the ones of OS, MA and WA, respectively.

For the initialization of OS, two rules are commonly used in the existing literature. The first one is random rule, the benefit of which is simplicity. The other one is the most number of operations remaining rule (Pezzella et al. 2008), which means the unprocessed job with more remaining operations

Table 1 Processing time table

Operations	M_1	M_2	M_3	M_4
J_1				
O_{11}	10	7	6	13
O_{12}	4	5	8	12
O_{13}	9	5	6	12
O_{14}	7	8	4	10
J_2				
O_{21}	15	12	8	6
O_{22}	9	5	7	13
O_{23}	14	13	14	20
J_3				
O_{31}	7	16	5	11
O_{32}	9	16	8	11
O_{33}	6	14	8	18

Table 2 Worker machine informatino matrix

	M_1	M_2	M_3	M_4
W_1	1	1	–	–
W_2	–	1	1	–
W_3	–	–	1	1

* "1" and "–" mean the worker is able or unable to use the machine respectively

Table 3 Coding scheme of a particle

OS	1	2	1	1	2	3	2	1	3	3
MA	4	1	3	2	1	4	3	2	1	2
WA	3	1	2	2	1	3	1	1	1	2

have higher priority. The benefit of the most number of operations remaining rule is the speed of convergence.

For the initialization of MA, the modified Approach by Localization and Earliest Finish Time rule are presented.

Approach by Localization is proposed by Kacem in (Kacem and Hammadi 2002), which allocates operations to the machine with minimal workload. However, this approach is strongly dependent on the order of operations. Here we slightly modify it to be more flexible. The Modified Approach by Localization is carried out as follows: firstly, rearrange the processing time table according to the mixed initialized OS, then the MA can be determined by using Approach by Localization. For example, with Table 1 and the OS in Table 3, we can obtain the rearranged operation sequence, shown as the first column of Table 4. The corresponding processing time of these operations on machine M_1–M_4 can be obtained in the Part1 of Table 4. Based on Approach by Localization, O_{11} should be processed on M_3 since 6 is the minimal processing time. Then update the processing time of other operations on M_3 we obtain Part2. It can be seen that O_{21} should be processed on M_4 because 6 is the minimal value. Repeat the process until the last operation finished, then we can obtain the corresponding MA vector (3, 4, 1, 2, 2, 1, 3, 4, 1, 2).

The key idea of Earliest Finish Time is taking into account the processing time and the earliest started time of resources on basis of assigned operations.

For the initialization of WA, random rule and Earliest Finish Time are used. The rules are just the same as the ones in OS and MA, respectively.

In this paper, the proportions of each rule for all the three chromosomes are 50 and 50 %, respectively.

Global search structure

The global search is carried out by updating the position of particles. The position updating method of DPSO was presented to deal with flow job shop in (Pan et al. 2008). DPSO updates the particles directly in discrete domain, which enhance the search efficiency. We now further extend the method to DRC-FJSSP. The position of the particle is updated as follows:

$$X_i^{k+1} = c_2 \otimes f_3 \left\{ \left[c_1 \otimes f_2 \left(w \otimes f_1 \left(X_i^k \right), pB_i^k \right) \right], gB^k \right\} \tag{13}$$

where X_i^k is the ith particle at kth generation, w is the inertia weight, c_1 and c_2 are acceleration coefficients between [0, 1]; pB_i^k and gB^k are the personal best and global best position in the swarm population at kth generation; f_1, f_2 and f_3 are operators. The updating process contains E_i^k, F_i^k and X_i^k, and they are formulated as follows:

$$E_i^k = w \otimes f_1 \left(X_i^k \right) = \begin{cases} f_1 \left(X_i^k \right); & r < w \\ X_i^k; & otherwise \end{cases} \tag{14}$$

where r is a uniform random number between [0,1]. If $r < w$, then $f_1(X_i^k)$ is carried out. First, randomly choose two different operations and exchange them. MA and WA are adjusted correspondingly to keep the allocations of machine and worker unchanged. Second, randomly generate two integers i and j between 1 and L, then replace the ith machine of MA and the jth worker of WA with another machine and worker randomly chosen from the candidate machine and worker set, respectively.

$$F_i^k = c_1 \otimes f_2 \left(E_i^k, pB_i^k \right) = \begin{cases} f_2 \left(E_i^k, pB_i^k \right); & r < c_1 \\ E_i^k; & otherwise \end{cases} \tag{15}$$

f_2 is the crossover operation for E_i^k and pB_i^k, which has a Precedence Preserving Order based Crossover (POX) oper-

Table 4 Modified approach by localization

	Part1				Part2				Part3				…	Part10			
	M_1	M_2	M_3	M_4	M_1	M_2	M_3	M_4	M_1	M_2	M_3	M_4	…	M_1	M_2	M_3	M_4
O_{11}	10	7	*6*	13	10	7	*6*	13	10	7	*6*	13	…	10	7	*6*	13
O_{21}	15	12	8	6	15	12	14	*6*	15	12	14	*6*	…	15	12	8	*6*
O_{12}	4	5	8	12	4	5	14	12	*4*	5	14	18	…	*4*	5	8	12
O_{13}	9	5	6	12	9	5	12	12	9	5	12	18	…	9	*5*	6	12
O_{22}	9	5	7	13	9	5	13	13	9	5	13	19	…	9	*5*	7	13
O_{31}	7	16	5	11	7	16	11	11	7	16	11	17	…	*7*	16	5	11
O_{23}	14	13	14	20	14	13	20	20	14	13	20	26	…	14	13	*14*	20
O_{14}	7	8	4	10	7	8	10	10	7	8	10	16	…	7	8	4	*10*
O_{32}	9	16	8	11	9	16	14	11	9	16	14	17	…	*9*	16	8	11
O_{33}	6	14	8	18	6	14	14	18	6	14	14	24	…	6	*14*	8	18

Italic values indicate the chosen machines for the corresponding operations

ation for OS, and two Rand-point Preservation Crossover (RPX) operations for MA and WA, respectively.

If $r < c_1$, then POX is carried out for OS. Randomly generate two nonempty job set J_1 and J_2, where $J_1 \cup J_2 = J$. Set $i = 1$ and check whether the ith operation in E_i^k belongs to J_1 or not. If yes, then copy it to F_i^k. If no, then check whether the ith operation in pB_i^k belongs to J_2 or not. If yes, then copy to F_i^k. If no, repeat the above procedure by setting $i = i+1$. After repeating L times, F_i^k is obtained. An example is illustrated in Fig. 1, where $J_1 = \{2\}$, $J_2 = \{1, 3\}$.

For the RPX of MA, first generate a vector $R = [r_1, r_2, \ldots, r_L]$, where $r_i \in (0, 1)$ are random scalars for $i = 1, 2, \ldots, L$. Record the indexes of r_i if $r_i \le pf$, then find the corresponding operations in pB_i^k and copy the corresponding elements of MA in pB_i^k to the elements of MA in F_i^k. The self-adaption probabilities pf is chosen as follows

$$pf = pf_{\max} - \frac{pf_{\max} - pf_{\min}}{iter} \times gen \tag{16}$$

where $iter$ is length of iteration, gen is the running time, $pf_{\max}$ and $pf_{\min}$ are the maximal and minimal self-adaption probabilities. The RPX of WA is the same as MA. An example is shown in Fig. 2, where $pf = 0.6$ is calculated by (16).

$$X_i^k = c_2 \otimes f_3\left(F_i^k, gB^k\right) = \begin{cases} f_3\left(F_i^k, gB^k\right); & r < c_2 \\ F_i^k; & otherwise \end{cases} \tag{17}$$

f_3 is a crossover operation for F_i^k and gB^k, which has two RPX for MA and WA, respectively, and keeps OS unchanged.

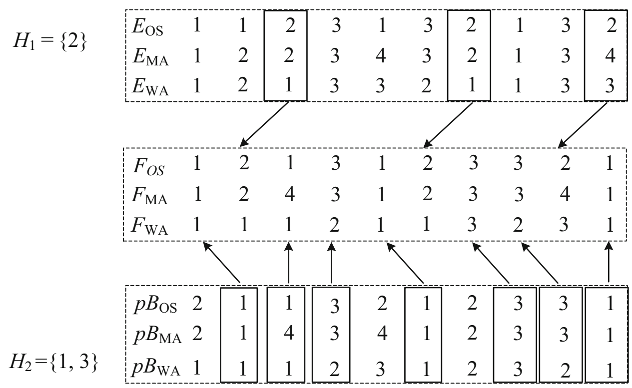

Fig. 1 POX crossover procedure

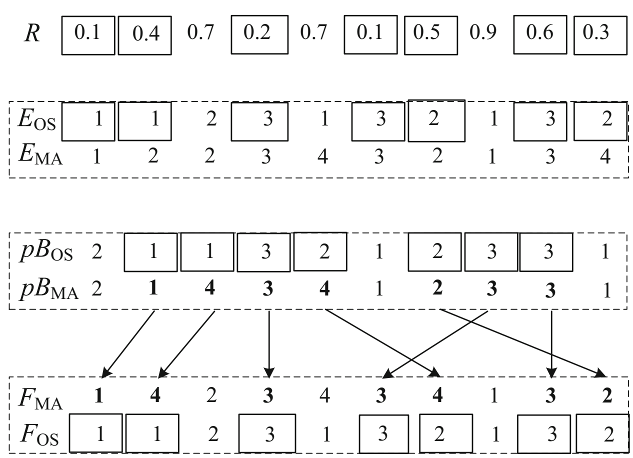

Fig. 2 RPX crossover procedure

Local search for HDPSO algorithm

SA is an effective local search algorithm which allows occasional alterations that worsen the solution in an attempt to increase the probability of leaving local optimum (Kirkpatrick et al. 1983). In SA, the choice of neighborhood can significantly influences algorithm performance.

In this subsection an ISA technique which has variable neighborhood structures is used as the local search of

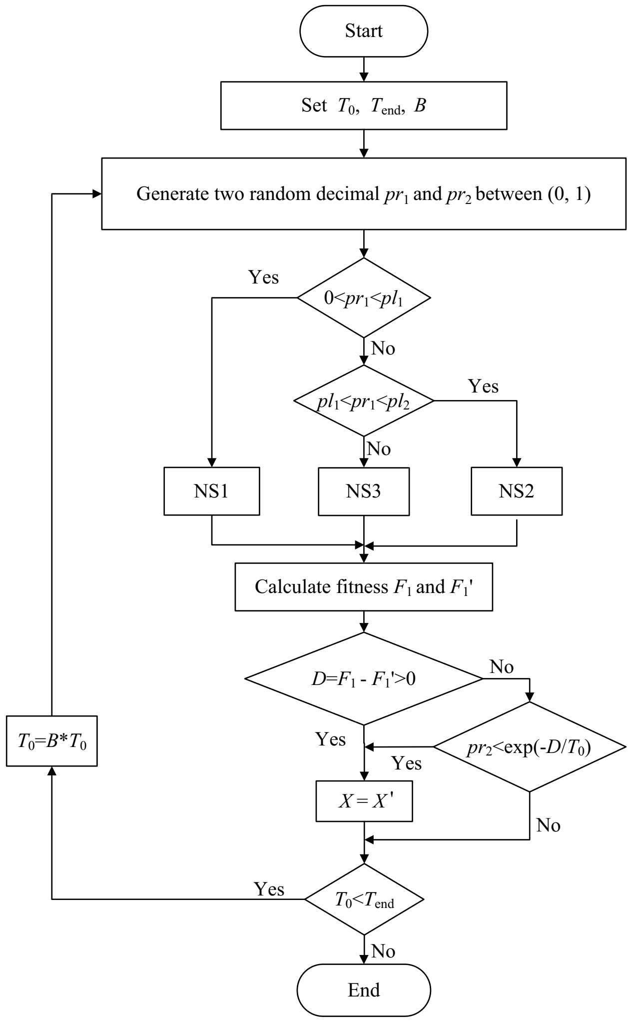

Fig. 3 Flow chart of the local search ISA

HDPSO. The flow chart of the local search ISA is given in Fig. 3, where T_0, T_{end} and B are initial temperature, final temperature, and annealing rate, respectively. Generate two random variables pr_1 and pr_2, where $pr_1, pr_2 \in (0, 1)$. pr_1 determines which neighborhood structure is used. pr_2 is the probability whether to accept a worse solution or not. $pl_1, pl_2 \in (0, 1)$ are parameters to be chosen. The main feature of ISA is to randomly choose a neighborhood from NS_1, NS_2 and NS_3 when carrying out simulating annealing.

NS_1 for OS

If $pr_1 < pl_1$, then neighborhood structure NS_1 is carried out to change OS based on the critical path. The neighborhood structure based on critical path has been introduced to scheduling problem successfully (Li et al. 2011a, b). This method always finds the feasible solution and refined neighborhoods by interchanging under given regulations, but is sensitivity to local convergence (Li et al. 2011a, b). So a global neighborhood structure is used in the paper: randomly choose a critical path $CP = \{cp_1, cp_2 \ldots cp_e\}$, where cp_e is a critical operation, e is the total operation number of *CP*. Randomly generate two integers x and y, where $x, y \in [1, e], x \neq y$, then inserting an critical operation cp_x to the front of the critical operation cp_y.

NS_2 for MA

If $pl_1 \le pr_1 < pl_2$, then neighborhood structure NS_2 is carried out. To generate neighboring solution, NS_2 changes the assignment of the operations to the machines (Yazdani et al. 2010), which only adjust MA of the particle. NS_2 is applied as follows:

Step 1 Calculate the workload of each machine and sequence them by the workload. Denote the machine sets by $\bar{M}_1, \bar{M}_2, \ldots, \bar{M}_{\max}$, respectively, where $\bar{M}_1$ and $\bar{M}_{\max}$ are the sets with minimal and maximal workload, respectively. Set $i = 1$.
Step 2 Randomly choose a machine M_a from $\bar{M}_{\max}$, then check whether there are operations that can be operated on either any machine of $\bar{M}_i$ or M_a.
Step 3 If yes, then denote the operations set by $\bar{O}_a$, randomly choose an operation O_c from $\bar{O}_a$ and the operating machine is denoted by M_b. Assign O_c from M_a to M_b. If not, then set $i = i + 1$ and go back to step2 until $i = \max - 1$.

NS_3 for WA

If $pr_1 \ge pl_2$, then neighborhood structure NS_3 is carried out. NS_3 follows the idea of NS_2 and changes WA of the particle. The procedure of NS_3 is similar to NS_2 and thus is omitted here for simplicity.

Procedure of HDPSO

Step 1 Set parameters swarm size P, maximum of generation G, w, c_1, c_2, T_0, T_{end}, B, pl_1 and pl_2.
Step 2 Set $k = 0$. Generate initial particle swarm $X^0 = \{X_1^0, X_2^0, \ldots, X_P^0\}$, initialize gB^k and pB_i^k.
Step 3 Let $k = k+1$, then update the current particle swarm $X^k = \{X_1^k, X_2^k, \ldots, X_P^k\}$ by DPSO. Update gB^k and pB_i^k.
Step 4 Choose $r \in [1, P/2]$ particles in the current population, executing local search based on ISA. Update gB^k and pB_i^k.
Step 5 If $k \le G$, then turn to Step 3, otherwise stop the loop and output the solutions.

Experiment results

In this section, several experiments are designed to test the performance of the proposed HDPSO. The experiments are implemented on a PC with VC++ 6.0, 2.0 GB RAM and 2.0 GHz CPU.

Table 5 Descriptions of instances R1–R10

Instances	*n*	*m*	*b*	*nob*	*sumop*
R1	5	8	5	[7, 9]	40
R2	6	8	5	[3, 8]	31
R3	8	10	7	[2, 8]	48
R4	9	7	4	[2, 10]	64
R5	10	5	3	[2, 9]	65
R6	10	6	4	[4, 10]	71
R7	14	5	3	[5, 9]	88
R8	14	8	5	[2, 10]	98
R9	15	7	4	[3, 10]	103
R10	15	9	6	[2, 10]	40

nob is the value range of operation number per job, *sumop* is the sum of all operations

Table 6 The experimental results of parameter combination

Parameters evaluation	*w*		*c*	
	S_i^w	$\bar{S}_i^w$	S_j^c	$\bar{S}_j^c$
Level 1	155,018	172.24	159,562	177.29
Level 2	155,658	172.95	159,167	176.85
Level 3	156,210	173.57	158,246	175.83
Level 4	156,903	174.34	157,817	175.35
Level 5	157,754	175.28	157,376	174.86
Level 6	158,040	175.60	156,637	174.04
Level 7	158,479	176.09	156,265	173.63
Level 8	158,613	176.24	155,784	173.09
Level 9	159,012	176.68	155,569	172.85
Range value	S'_w	4.44	S'_c	4.44

Preparation

First, instances R1–R10 are randomly generated with parameters shown in Table 5 and they are also subject to the following constraints: (a) the processing time of each job is t_{pqh}, where t_{pqh} is a random integer between [2,20]. (b) every operation can be processed on two different machines. (c) each worker is able to operate i machines, where i is a random integer between [1,3]. Then, these instances R1–R10 are used to analyze the parameters and test the validity of ISA.

Finally, two collections of benchmark instances are used to compare the proposed HDPSO with other algorithms. One collection is four classic FJSSP $n \times m$ benchmark instances (Kacem and Hammadi 2002) with only machine resource constraint, where the dimension of Q1, Q2, Q3 and Q4 are 4×5, 8×8, 10×7 and 10×10, respectively. The other collection is $n \times m \times b$ instances C1 (Ju and Zhu 2006), C2 (ElMaraghy et al. 2000), C3 (Cao and Yang 2011) and C4 (Liang and Tao 2011) for DRC-FJSSP where the dimension of C1, C2, C3 and C4 are $4 \times 6 \times 4$, $4 \times 10 \times 7$, $6 \times 6 \times 4$ and $10 \times 6 \times 4$, respectively.

Since the parameters to be tested are the inertia weights w and learning factors c, thus the other parameters are chosen according to the way in (Xia and Wu 2005; Zhang et al. 2012) and then through various experimental tests. The parameters are chosen as follows:

$$T_0 = 3,\ T_{\text{end}} = 0.01,\ B = 0.9,\ pl_1 = 0.6,\ pl_2 = 0.8,$$
$$pf_{\max} = 0.9,\ pf_{\min} = 0.2,\ P = 100,\ G = 100.$$

Table 7 The results of HDPSO and DPSO with the same iterations $G = 100$

Instances	HDPSO					DPSO				
	ARE	BF	AF	WF	AT	ARE	BF	AF	WF	AT
R1	0.02	97	99.1	102	3.9	0.09	101	106.1	112	0.3
R2	0.09	71	77.6	84	3.8	0.23	80	87.6	95	0.3
R3	0.07	94	100.7	109	4.1	0.12	98	105.5	115	0.5
R4	0.04	148	154.4	162	6.6	0.09	152	161.7	171	0.7
R5	0.07	200	213	225	7.1	0.12	211	223.3	236	1.1
R6	0.06	179	189.1	199	6.6	0.09	185	194.9	203	1
R7	0.04	259	269.7	281	7.4	0.06	268	275.8	290	1.6
R8	0.08	190	204.9	214	5.9	0.09	202	207.3	215	1.7
R9	0.05	236	247	258	6.6	0.08	250	254.7	268	2
R10	0.04	138	144.2	155	5.4	0.09	146	150	156	1.5

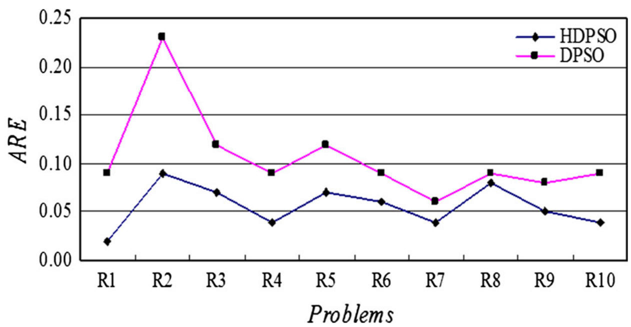

Fig. 4 The ARE comparison chart of HDPSO and DPSO with the same iterations $G = 100$

Parameters analysis

In HDPSO, intertia weights w and learning factors c_1 and c_2 have key impact. It is chosen that $c_1 = c_2$ in general and denoted by c. In order to obtain the appropriate values for w and c, we solve R1–R10 by various values of w and c. Nine values are chosen for both w and c, and they are 0.1, 0.2, …,0.9, respectively. The corresponding objective F with each combination of w and c running 10 times on all instances R1–R10 can be obtained by (1)–(12). The obtained makespan are denoted by $F_{ijk}, i, j = 1, 2, \ldots, 9; k = 1, 2, \ldots, 10$, where i, j represent the value of w and c, and k represents index of the instance R1–R10, for example $i = 1, j = 1$ and $k = 1$ mean $w = 0.1, c = 0.1$ and instance R1, respectively.

Then the range analysis method (Xia 1985) is used for the analysis and the result is shown in Table 6. The evaluation value of two parameters are defined as $S_i^w = \sum_{j=1}^{9} \sum_{k=1}^{10} F_{ijk}, \quad i = 1, 2, \ldots, 9$ and $S_j^c = \sum_{i=1}^{9} \sum_{k=1}^{10} F_{ijk}, j = 1, 2, \ldots, 9$, respectively, where the upper and the lower index of S_i^w or S_j^c are the parameter and the level, respectively. Another evaluation values $\bar{S}_i^w$ and $\bar{S}_j^c$, are defined as $\bar{S}_i^w = S_i^w/900$ and $\bar{S}_j^c = S_j^c/900$, respectively.

Table 9 Results of the four Kacem instances

Instances	AL+CGA	MOPSO+LS	ABC	HDPSO
Q1	16	16	11	11
Q2	15	15	14	14
Q3	15	–	11	11
Q4	7	7	7	7

The range value is defined as $S'_w = \max S_i^w - \min S_i^w$ and $S'_c = \max S_j^c - \min S_j^c$.

From Table 6, we can see $\bar{S}_i^w$ achieves the minimal value 172.24 on level 1 for parameter w, and $\bar{S}_j^c$ achieves the minimal value 172.85 on level 9 for parameter c. Then it can be seen in this test $w = 0.1, c = 0.9$ is the best parameter combination. In the following experiments, w and c are chosen as 0.1 and 0.9, respectively. As for the range value, we can see that the two parameters have the same impact to the results by $S'_w = S'_c = 4.44$.

The validity test of ISA

In this subsection, the validity of introducing ISA is tested by comparing HDPSO with DPSO. R1–R10 randomly run 10 times with iterations $G = 100$ and results are shown at Table 7, where $ARE = (AF - C^*)/C^* \times 100\,\%$ is the average relative error, C^* is the best solution with the two algorithms, *BF* is the best fitness, *AF* is the average value of 10 runs, *WF* is the worst fitness, and *AT* is the running time.

It can be clearly seen from Table 7 that HDPSO achieve better *BF* and *WF* on all the instances, which means HDPSO has stronger ability to get the better solution compared with DPSO. As can be seen in Fig. 4, HDPSO obtained smaller *ARE* than DPSO, indicating HDPSO is more stable. However, the drawback of HDPSO is that the runtime is longer than DPSO, showing HDPSO has a little more time consuming.

Table 8 Results of HDPSO and DPSO with the same runtime

Instances	HDPSO					DPSO				
	ARE	BF	AF	WF	AG	ARE	BF	AF	WF	AG
R1	0.07	91	97.6	106	38	0.13	99	102.9	107	84
R2	0.04	75	78.1	84	20	0.12	78	83.7	89	251
R3	0.09	94	102.5	116	71	0.15	100	108	113	81
R4	0.04	148	154.5	164	44	0.04	148	154	160	281
R5	0.07	196	210.6	222	54	0.12	214	219	224	129
R6	0.06	172	186.2	195	28	0.04	177	183.2	191	103
R7	0.05	258	271.3	284	39	0.09	270	280.5	291	43
R8	0.06	191	203	213	41	0.09	200	207.6	215	62
R9	0.07	234	249.4	262	41	0.08	249	253	258	52
R10	0.07	136	145.5	155	47	0.05	140	147.4	156	63

Table 10 Results of the four DRC FJSSP instances

Instances	C1		C2		C3		C4	
Algorithm	GA	HDPSO	GAs	HDPSO	IGA_2	HPAO	GATS	HDPSO
Results	17	17	63	50	35	35	55	46

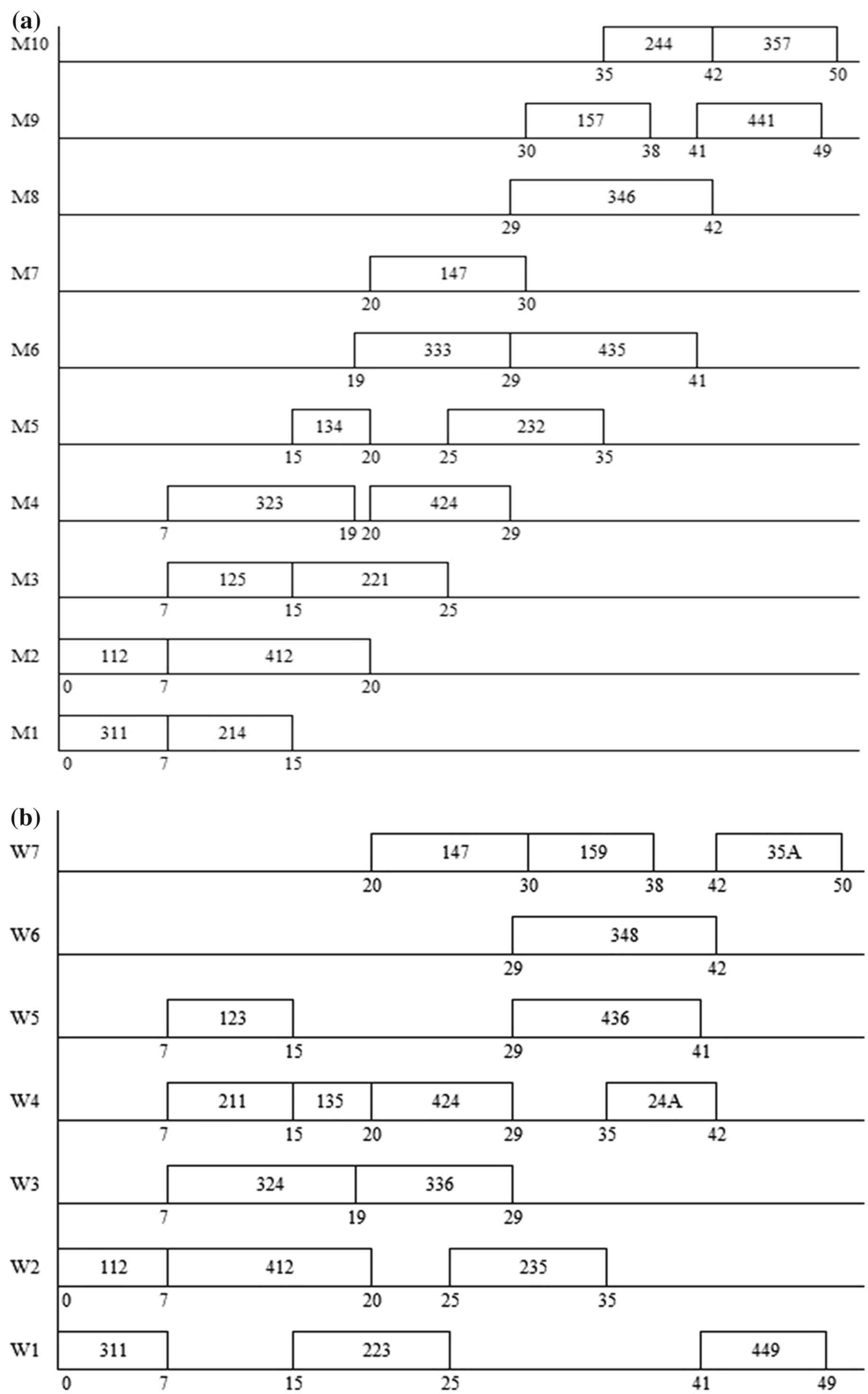

Fig. 5 **a** Machine Gantt chart of problem C2 ($F = 50$). **b** Worker Gantt chart of problem C2 ($F = 50$)

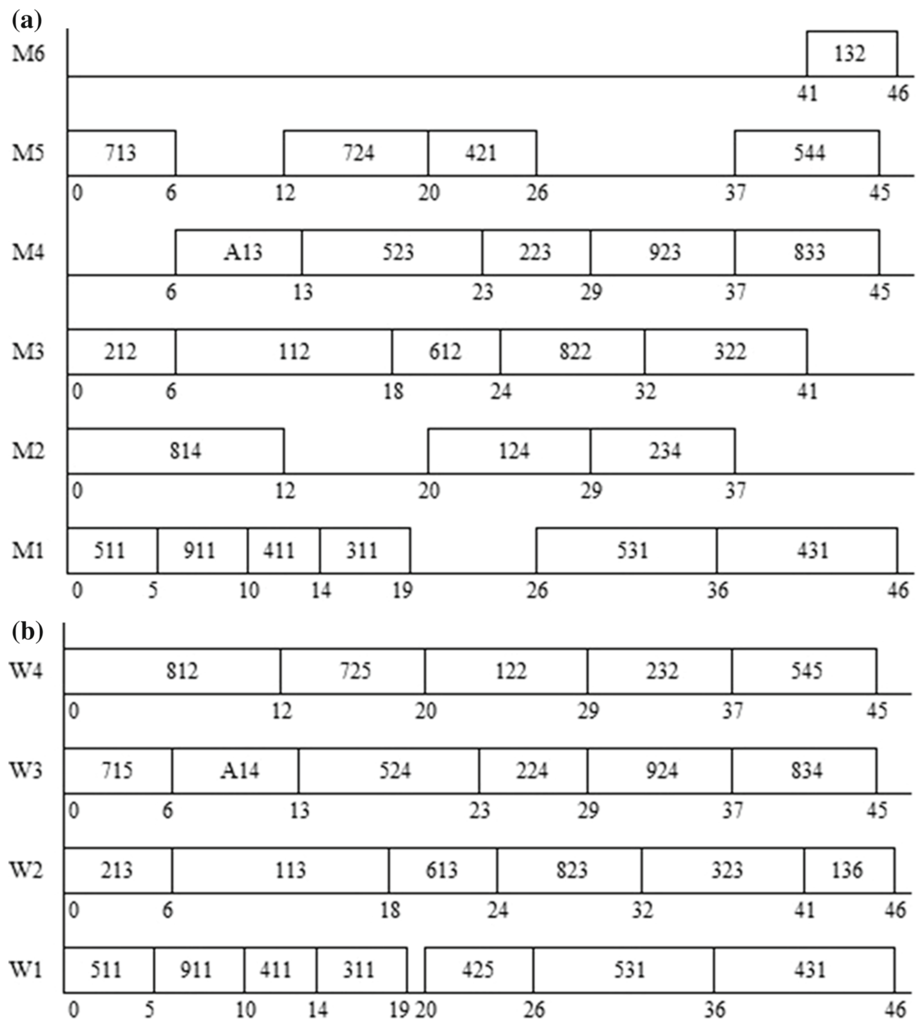

Fig. 6 **a** Machine Gantt chart of problem C4 ($F = 46$). **b** Worker Gantt chart of problem C4 ($F = 46$)

The performance of HDPSO and DPSO are tested with the same runtime. The runtime is chosen to be 5 s. Table 8 shows the experimental results, from which we can see HDPSO achieves better *ARE*, *BF*, *AF* and *WF* on instances R1–R3, R5, R7–R10. As for the *AG*, HDPSO is better on all instances than DPSO, which means the convergence speed is faster. That is to say, even in a time-critical setting, HDPSO is still more effective than DPSO.

Thus, it can be concluded from the experimental results in this subsection that ISA enhance the ability of both exploitation and exploration besides adding some computational time in solving the DRC-FJSSP.

Results comparisons

In this subsection, two sets of benchmark examples are employed to compare the proposed HDPSO with other algorithms. The first set of examples is for single-resource constraint problems and the other set is for DRC-FJSSP.

Single resource constraint problems

Table 9 shows the results of our approach HDPSO compared with the algorithms of AL+CGA (Kacem and Hammadi 2002), MOPSO+LS (Moslehi and Mahnam 2011) and ABC (Wang et al. 2012). We can see our approach can find the optimal solution for all the four Kacem instances (Kacem and Hammadi 2002). So the proposed approach is also effective to solve single-resource constraint job scheduling problems.

DRC-FJSSP

For DRC-FJSSP, C1 is used to compared with GA (Ju and Zhu 2006), C2 is used to compared with GAs (ElMaraghy et al. 2000), C3 is used to compared with IGA$_2$ (Cao and Yang 2011), and C4 is used to compared with GATS (Liang and Tao 2011). The results are showed in Table 10. For instances C1 and C3, the optimal values are the same as the best solutions of the other algorithms.

For instances C2 and C4, the optimal solution of HDPSO is better than the best solutions of other algorithms. Figure 5a, b give the Gantt chart of the optimal solution for instance C2 obtained by HDPSO and Fig. 6a, b give the Gantt chart of the optimal solution for instance C4 obtained by HDPSO. M and W stands for machine and worker, respectively. The number following M or W is the index of the resource number. The 3-digit number in the blocks are job number, operation number and resource number (machine number or worker number) respectively. "A" stands for the number 10.

Conclusions

This paper solved the DRC-FJSSP by presenting a HDPSO algorithm with improved SA technique. The main benefits of this algorithm include the exclusion of infeasible solutions and improvements of the local search ability. Experimental results showed the effectiveness of the proposed method and the superiority over some available results.

The computation time is a critical issue that limits the application of this method. Moreover, some parameters in the algorithm are chosen not in a systemic way. How to choose more appropriate parameters for the algorithm is also helpful in improving the performance. Since other objectives such as production cost and total overload are also important in the manufacturing systems, multi-objective case for the considered problem is an interesting and important topic that deserves investigation.

Acknowledgements This project is supported by National Natural Science Foundation of China (NSFC Grant No. 61379123) and the National Key Technology R&D Program in the 12th Five Year Plan of China (Grant No. 2012BAD10B01).

References

Adibi, M. A., & Shahrabi, J. (2014). A clustering-based modified variable neighborhood search algorithm for a dynamic job shop scheduling problem. *International Journal of Advanced Manufacturing Technology*, *70*(9–12), 1955–1961.

AitZai, A., Benmedjdoub, B., & Boudhar, M. (2014). Branch-and-bound and PSO algorithms for no-wait job shop scheduling. *Journal of Intelligent Manufacturing*. doi:10.1007/s10845-014-0906-7

Cao, X. Z., & Yang, Z. H. (2011). An improved genetic algorithm for dual-resource constrained flexible job shop scheduling. In *International conference on intelligent computation technology and automation, Shen Zhen*.

Coello, C. A. C., Pulido, G. T., & Lechuga, M. S. (2004). Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, *8*(3), 256–279.

ElMaraghy, H., Patel, V., & Ben Abdallah, I. (1999). A genetic algorithm based approach for scheduling of dual-resource constrainded manufacturing systems. *Journal of Manufacturing Systems*, *48*(1), 369–372.

ElMaraghy, H., Patel, V., & Ben Abdallah, I. (2000). Scheduling of manufacturing systems under dual-resource constraints using genetic algorithms. *Journal of Manufacturing Systems*, *19*(3), 186–201.

Gargeya, V. B., & Deane, R. H. (1996). Scheduling research in multiple resource constrained job shop: A review and critique. *International Journal of Production Research*, *8*(34), 2077–2097.

González, M., Vela, C., González-Rodríguez, I., & Varela, R. (2013). Lateness minimization with Tabu search for job shop scheduling problem with sequence dependent setup times. *Journal of Intelligent Manufacturing*, *24*(4), 741–754.

Ju, Q. Y., & Zhu, J. Y. (2006). Study of fuzzy job shop scheduling problems with dualresource and multi-process routes. *Mechanical Science and Technology*, *12*, 1424–1427.

Kacem, I., & Hammadi, S. (2002). Approach by localization and multi-objective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Transaction on Systems, Man, and Cybernetics Part C: Applications and Reviews*, *32*(1), 1–13.

Katherasan, D., Elias, J., Sathiya, P., & Haq, A. N. (2014). Simulation and parameter optimization of flux cored arc welding using artificial neural network and particle swarm optimization algorithm. *Journal of Intelligent Manufacturing*, *25*(1), 67–76.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of the 4th IEEE international conference on neural networks, Piscataway*.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671–680.

Lei, D. M. (2010). A genetic algorithm for flexible job shop scheduling with fuzzy processing time. *International Journal of Production Research*, *48*(10), 2995–3013.

Lei, D. M., & Guo, X. P. (2014). Variable neighbourhood search for dual-resource constrained flexible job shop scheduling. *International Journal of Production Research*, *52*(9), 2519–2529.

Li, J. Y., Sun, S. D., Huang, Y., & Wang, N. (2010). Research into self-adaptive hybrid ant colony algorithm based on flow control. In *The 2nd international workshop on intelligent systems and applications, Wuhan*.

Li, J. Q., Pan, Q. K., Suganthan, P. N., & Chua, T. J. (2011). A hybrid tabu search algorithm with an efficient neighborhood structure for the flexible job shop scheduling problem. *International Journal of Advanced Manufacturing Technology*, *52*(5–8), 683–697.

Li, J. Y., Sun, S. D., & Huang, Y. (2011). Adaptive hybrid ant colony optimization for solving dual resource constrained job shop scheduling problem. *Journal of Software*, *6*(4), 584–594.

Liang, D., & Tao, Z. (2011). Hybrid genetic-Tabu Search approach to scheduling optimization for dual-resource constrained job shop. In *Cross strait quad-regional radio science and wireless technology conference, Harbin*.

Lin, T.-L., Horng, S.-J., Kao, T.-W., Chen, Y.-H., Run, R.-S., Chen, R.-J., et al. (2010). An efficient job-shop scheduling algorithm based on particle swarm optimization. *Expert Systems with Applications*, *37*(3), 2629–2636.

Moslehi, G., & Mahnam, M. (2011). A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. *International Journal of Production Economics*, *129*(1), 14–22.

Nie, L., Gao, L., Li, P., & Li, X. (2013). A GEP-based reactive scheduling policies constructing approach for dynamic flexible job shop scheduling problem with job release dates. *Journal of Intelligent Manufacturing*, *24*(4), 763–774.

Pan, Q. K., Tasgetiren, M. F., & Liang, Y. C. (2008). A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. *Computers & Operations Research*, *35*(9), 2807–2839.

Pérez, M. F., & Raupp, F. P. (2014). A Newton-based heuristic algorithm for multi-objective flexible job-shop scheduling problem. *Journal of Intelligent Manufacturing*. doi:10.1007/s10845-014-0872-0

Pezzella, F., Morganti, G., & Ciaschetti, G. (2008). A genetic algorithm for the flexible job-shop scheduling problem. *Computers & Operations Research*, *35*(10), 3202–3212.
Qiu, X., & Lau, H. K. (2014). An AIS-based hybrid algorithm for static job shop scheduling problem. *Journal of Intelligent Manufacturing*, *25*(3), 489–503.
Vijay Chakaravarthy, G., Marimuthu, S., & Naveen Sait, A. (2013). Performance evaluation of proposed differential evolution and particle swarm optimization algorithms for scheduling m-machine flow shops with lot streaming. *Journal of Intelligent Manufacturing*, *24*(1), 175–191.
Wang, L., Zhou, G., Xu, Y., Wang, S. Y., & Liu, M. (2012). An effective artificial bee colony algorithm for the flexible job-shop scheduling problem. *International Journal of Advanced Manufacturing Technology*, *60*(1–4), 303–315.
Xia, B. Z. (1985). *Orthogonal experiment*. Changchun: Jilin People Publishing House.
Xia, W. J., & Wu, Z. M. (2005). An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering*, *48*(2), 409–425.
Xu, J., Xu, X., & Xie, S. Q. (2011). Recent developments in dual resource constrained (DRC) system research. *European Journal of Operational Research*, *215*(2), 309–318.
Yazdani, M., Amiri, M., & Zandieh, M. (2010). Flexible job-shop scheduling with parallel variable neighborhood search algorithm. *Expert Systems with Applications*, *37*(1), 678–687.
Zhang, H., Li, H., & Tam, C. M. (2006). Particle swarm optimization for resource-constrained project scheduling. *International Journal of Project Management*, *24*(1), 83–92.
Zhang, R., & Wu, C. (2010). A divide-and-conquer strategy with particle swarm optimization for the job shop scheduling problem. *Engineering Optmization*, *42*(7), 641–670.
Zhang, J., Wang, W. L., & Xu, X. L. (2012). Hybrid particle-swarm optimization for multi-objective flexible job-shop scheduling problem. *Control Theory & Applications*, *29*(6), 715–722.