2021 5th Asian Conference on Artificial Intelligence Technology (ACAIT) | 978-1-6654-2630-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/ACAIT53529.2021.9731228

Adapting a reinforcement learning method for the distributed blocking hybrid flow shop scheduling problem

Haoxiang Qin
School of Computer Science
Liaocheng University
Liaocheng, China
987352978@qq.com

Yuting Wang✉
School of Computer Science
Liaocheng University
Liaocheng, China
wangyuting@lcu-cs.com

Yuyan Han✉
School of Computer Science
Liaocheng University
Liaocheng, China
hanyuyan@lcu-cs.com

Qingda Chen
State Key Laboratory of Synthetical Automation for Process Industries
Northeastern University
Shenyang, China
cqd0309@126.com

Junqing Li
School of information science and Engineering
Shandong Normal University
Jinan, China
lijunqing@lcu-cs.com

***Abstract*—With the continuous emission of energy in the past years, the environmental problems are becoming more and more serious. For example, in manufacturing, the energy efficient scheduling problem has become particularly prominent, and attracted much attention of the researchers. As a common scheduling problem in the real world, the research on distributed blocking hybrid flow shop (DBHFSP) is very few. In this paper, we will carry out a study of the problem. Because of its NP-hard character, therefore, we use the intelligent optimization algorithm to solve the problem. we firstly introduce the MILP model of the DBHFSP, then, the Q-learning method combined with the IG algorithm framework (IGQ) is proposed to solve this problem. In the experimental part, through the experiment results and comparison with other algorithms in the recent literatures, the proposed algorithm shows the excellent performance in the simulation experiment with the objective of minimizing the energy consumption.**

***Index Terms*—DBHFSP, Q-learning method, IG algorithm, energy-efficient**

I. Introduction

The hybrid flow shop scheduling problem (HFSP), as an extent of the traditional flow shop scheduling problem (FSP), has been researched by many people [1]. In HFSP, there are a set of stages in the processing plant, in the plant, it has a number of unrelated parallel machines, a series of jobs must pass all the stages. In recent years, due to the applicability of HFSP, many people have obtained many positive result on HFSP.

With the development of economic globalization, the production mechanism of a single factory is difficult to meet the needs of the current market, therefore, in order to improve the production efficiency of enterprises to meet the needs of the market quickly. Most enterprises adopt the multi-factory production mechanism, that is, the distributed flow shop scheduling problem (DPFSP) [2]. Compared with the traditional FSP, DPFSP can allocate resources more efficiently and improve the production efficiency of enterprises. However, it is more complex than FSP because it also involves the allocation of jobs.

In DPFSP, for the single factory with parallel machines scheduling in each plant has attracted the attention of scholars in recent years [3]. In any processing factory, each factory contains the same steps of processing stage. For different processing stages, it is often assumed that there are infinite buffers between adjacent machines, and the job can be stored in these buffers until it is processed by the next stage machine. However, in the actual production of the enterprise, due to the limited storage space, there is no buffer between adjacent machines to store the jobs, so the blocking condition [4] of jobs in different machines should be considered. At this time, the problem becomes distributed blocking hybrid flow shop scheduling problem (DBHFSP). As far as the author knows, there is no corresponding research solving the DBHFSP, but the above situation is very common in real world, thus, it is necessary to study the problem.

Iterative Greedy (IG) algorithm is an intelligent optimization algorithm which contains a simple structure. Different from other intelligence algorithms, it only yields one solution in every iteration. IG was first used by Rubén to solve the FSP [5], in this paper, a new IG algorithm which contains a reinforcement learning (RL) method is developed to reduce the energy of DBHFSP.

National Natural Science Foundation of China: No. 61773246, 61603169, 61803192, 61966012, 61773192, 61973203, and 71533001, and the Youth Innovation Talent Introduction and Education Program support received from Shandong province colleges and universities.

978-1-6654-2630-5/21/$31.00 ©2021 IEEE

Q-learning algorithm, is a basic method of RL, it has been applied to solve FSP [6]. It shows the cumulative reward of taking an action in some states by learning from the environment The core of the Q-learning algorithm is a simple iterative updating of values. Each state-action (s, a) has a related Q-value. In this paper, Q-learning mechanism is embedded into the IG algorithm as a selection strategy, namely IG-Q-learning (IGQ) algorithm.

The main contributions are given as follows:

(1)It is a simple, easy to implement and reduce the computational complexity of the original IG algorithm.

(2)The Q-learning method can balance the global search and local search ability of IGQ, and effectively reduce the energy consumption caused by blocking.

(3)Global and local search strategies effectively increase the diversity and convergence of IGQ, and find the near-optimal possible job sequence.

II. Distributed Blocking Hybrid Flow Shop Scheduling Problem

The DBHFSP consists of some identical factories, each factory includes the same processing stages. Each stage has two parallel machines. A series of jobs should be processed on one of these factories. No buffer exists between any two continuous stages. The problem is to allocate these jobs to one of these identical factories and determine the processing order in the same factory to minimize the energy consumption. In addition, the DBHFSP is subject to the following constrains.

1)Once the job is processed into one factory, it cannot be processed in other factories.

2)Each machine can process only one job at the time and each job can be processed on at one machine.

3)All jobs should be continuously processed not be pre-empted and interrupted.

4)No buffer exists between any two continuous stages.

5)Interruption and pre-emption of the processing jobs are not allowed.

6)Both setup and transportation time are included in the processing time.

According the MILP of the DHFSP [7], The mathematical model of DBHFSP is given as follows:

J: The number of jobs.

F: The number of factories.

S: The number of stages.

j: The index of the jobs, $j \in \{1, 2..., J\}$

f: The index of the factories, $f \in \{1, 2..., F\}$

s: The index of the stages, $s \in \{1, 2..., S\}$

m: The index of the machines at each stage, $m \in \{1, 2\}$

$m_{f,s}$: The first available machine at stage s in factory f under the current moment.

$p_{j,s}$: Processing time of job j at stage s

$EC_{f,s}^{Process}$: Energy consumption per unit time of a job which is processed at stage s in factory f

$EC_{f,s}^{Blocking}$: Energy consumption per unit time of a job which is blocked at stage s in factory f

$EC_{f,s}^{Idle}$: Energy consumption per unit time of a machine which is in idle state.

TEC: The total energy consumption.

PEC: The energy consumption that machines stay at the processing state.

BEC: The energy consumption that machines stay at the blocking state.

IEC: The energy consumption that machines stay at the idle state.

$B_{f,j,s}$: The beginning time of job j at stage s in factory f.

$C_{f,j,s}$: The completion time of job j at stage s in factory f.

U: A very large number.

$x_{f,j}$: Binary variable which equals to 1 if job j is assigned in factory f, 0 otherwise.

$y_{f,s,j,m}$: Binary variable which equals to 1 if job j is processed on machine m at stage s in factory f, 0 otherwise.

$z_{f,s,j,j'}$: Binary variable if equals 1 when job j is processed before job j' on stage s in factory f, 0 otherwise .

Objective:

$$MinTEC = PEC + BEC + IEC \tag{1}$$

$$PEC = \sum_{f=1}^{F}\sum_{s=1}^{S}\sum_{j=1}^{J} EC_{f,s}^{Process} \cdot p_{j,s} \cdot y_{f,s,j,m}$$
$$m = 1 || m = 2 \tag{2}$$

$$BEC = \sum_{f=1}^{F}\sum_{s=2}^{S}\sum_{j=1}^{J} EC_{f,s}^{Blocking} \cdot (m_{f,s} - C_{f,j,s-1}) \cdot x_{f,j}$$
$$m_{f,s} \geqslant C_{f,j,s-1} \tag{3}$$

$$IEC = \sum_{f=1}^{F}\sum_{s=2}^{S}\sum_{j=1}^{J} EC_{f,s}^{Idle} \cdot (C_{f,j,s-1} - m_{f,s}) \cdot x_{f,j}$$
$$m_{f,s} \leqslant C_{f,j,s-1} \tag{4}$$

$$s.t. \sum_{f=1}^{F} x_{f,j} = 1, \forall j \tag{5}$$

$$\sum_{m=1}^{2} y_{f,s,j,m} = x_{f,j}, \forall f, j, s \tag{6}$$

$$B_{f,j,1} \geqslant 0, \forall f, j \tag{7}$$

$$B_{f,j,s+1} \geqslant B_{f,j,s} + p_{j,s}, \forall f, j, s \tag{8}$$

$$z_{f,s,j,j'} + z_{f,s,j',j} \leqslant 1, \forall f, s, j, j' \tag{9}$$

$$z_{f,s,j,j'} + z_{f,s,j',j} \geqslant y_{f,s,j,m} + y_{f,s,j',m} - 1$$
$$\forall f, s, j > j', m = 1 || m = 2 \tag{10}$$

$$B_{f,j',s} - (B_{f,j,s} + p_{j,s}) + U \cdot (3 - y_{f,s,j,m} - y_{f,s,j',m} - z_{f,s,j,j'}) \geqslant 0 \quad \forall j \neq j', f, s, m \in \{1,2\} \tag{11}$$

$$C_{f,j,s} = B_{f,j,s} + p_{j,s}, \forall f, j, s \tag{12}$$

$$p_{j,s} > 0 \tag{13}$$

$$m_{f,s} \geqslant 0 \tag{14}$$

$$x_{f,j} \in \{0,1\}, \forall f, j \tag{15}$$

$$y_{f,s,j,m} \in \{0,1\}, \forall f, s, j, m = 1 || m = 2 \tag{16}$$

$$z_{f,s,j,j'} \in \{0,1\}, \forall f, s, j, j' \tag{17}$$

The objective (1) is to minimize the total energy consumption. Equations (2-3) indicate the energy consumption when jobs are processed and blocked. Equation (4) expresses the energy consumption when machine $m_{f,s}$ stays in idle state. Constraint (5) makes sure that each job can be assigned on one factory for processing at most, (6) makes sure that each job can be processed by only one machine at each stage. Constraint (7) makes sure that the beginning time of jobs are not less than 0, Constraint (8) makes sure that the job can be processed at the next stage only when it is finished at previous stage. Constraints (9-11) make sure that the machines can only process one job at one time. Constraint (12) makes sure that the relationship of start time and completion time, Constraint (13) makes sure that the job's processing time is greater than 0. Constraint (14) makes sure that the time of the available machine is greater than 0. Constraints (15-17) show the binary decision variables.

III. The Proposed Algorithm

A. The procedure of the IGQ

This part designs the IGQ method in detail and it contains three parts, initialization strategy, global search strategy, local search strategy. The framework of IGQ is given in Algorithm 1.

Algorithm 1 The framework of the IGQ Algorithm

Input: $\pi = \{\pi_1, \pi_2, ..., \pi_J\}$ all parameters used in this algorithm
Output: π^{best} and TEC
Begin:
  $\pi^{\text{temp}} = \pi$
Initialization:
  Using the NEH to assign the jobs to the F factories.
  GlobalSearchStrategy(π, π^{temp});
While the termination criterion is not satisfied **do**
local search strategy:
    Q-learning-method(π^{temp});
global search strategy:
    GlobalSearchStrategy(π, π^{temp});
    If π^{temp} better than π **then**
      $\pi = \pi^{\text{temp}}$
      If π better than π^{best} **then**
        $\pi^{\text{best}} = \pi$
      End If
    End If
End While
End

B. The initialization strategy

It can be seen from Algorithm 1 that IGQ algorithm always iterates one solution in the process. Thus, it is important to use an initialization strategy to minimize the TEC. Nawaz, Enscore, and Ham (NEH) [8] is a heuristic algorithm with superior performance and it has been applied to solve various FSP. Huang et al. [9] presented an algorithm, named NEH-F, which is on basis of the multiple factories for solving the DPFSP. This paper also uses the NEH-F to arrange jobs to these factories. The specific details of NEH-F heuristic are presented with Algorithm 2.

Algorithm 2 NEH-F heuristic

Input: $\pi = \{\pi_1, \pi_2, ..., \pi_J\}$
Output: π^{temp}
Begin:
  Computing the total processing time of all jobs
  Sorting these jobs in a descending order, denoted as π^{temp}
  For j=1 to F
    Take job π_j^{temp} from π^{temp} and arrange to factory j
  End For
  For j=F+1 to n
    For f=1 to F
      Extract the job π_j^{temp} from π^{temp},then insert into all the positions of factory f
      EC_f is the minimum energy consumption
      Pos_j is the best position
    End For
    $Pos_j^{best} = arg(min_{f=1}^{F} EC_f)$
    Insert π_j^{temp} to the position Pos_j^{best}
  End For
End

C. The global search strategy

The global search strategy designed in this paper can improve the diversity of solutions and reduce the energy waste caused by blocking constraints, The strategy is designed as follows.

Algorithm 3 Global search strategy

Input: π, π^{temp},bool value $flag$
Output: π^{temp}

Begin:
Computing the total energy consumption of π, denoted as EC_{old}
Find the factory $\pi^{\text{temp}-\text{f}_{\max}}$ that consumes the most energy
While flag == true
Randomly select a factor $\pi^{\text{temp}-\text{f}_{\text{random}}}$ which is different from $\pi^{\text{temp}-\text{f}_{\max}}$
For j=1 to n
Randomly select a job $\pi_j^{\text{temp}-\text{f}_{\text{random}}}$ from $\pi^{\text{temp}-\text{f}_{\text{random}}}$
Randomly select a job $\pi_j^{\text{temp}-\text{f}_{\max}}$ from $\pi^{\text{temp}-\text{f}_{\max}}$
Swap $\pi_j^{\text{temp}-\text{f}_{\text{random}}}$ and $\pi_j^{\text{temp}-\text{f}_{\max}}$
Computing the total energy consumption of π^{temp}, denoted as EC_{new}
If $EC_{old} < EC_{new}$
flag = true
$\pi = \pi^{\text{temp}}$
Else
$\pi^{\text{temp}} = \pi$
End If
End For
End While
If π^{temp} better than π **then**
$\pi = \pi^{\text{temp}}$
End If
End

D. The Q-learning method

For fitting the Q-learning method, it needs to set the states as the factories. The actions are set as the changes in the relations. An action step is performed by a selection mechanism, which select the strategies according to the fitness of each factory. At the beginning the selections are random. As learning proceeds, the Q-value table are updated, and it influences the action selection. In the Q-value table, rows (states) represent different factories, the columns (actions), representing different selection strategies, this paper proposes 5 selection strategies. Fitness is the reciprocal of the energy consumption. In addition, The Q-value update function can be expressed as $Q(s_t, a_t) = (1-\alpha)Q(s_t, a_t) + \alpha(r_{t+1} + \gamma maxQ(s_{t+1}, a_{t+1})$. The $Q(s_t, a_t)$ represents the Q value that take the actiona_t at state s_t, α shows the learning rate, γ shows the discount rate, r_{t+1} shows the reward value after taking the action a_t, in this paper, it indicates the difference value between the new and old fitness values. $Q(s_{t+1}, a_{t+1})$ indicates the expected Q value that take the action a_{t+1} at state s_{t+1}. Q-learning selection mechanism can also balance the diversity and convergence of solutions. The specific implementation steps are shown in algorithm 4.

Algorithm 4 The Q-learning method

Input: π^{temp}
Output: π^{temp}
Begin:
If it is the first time to select these strategies
For f = 1 to F
Choose a strategy i at random for the job sequence $\pi^{\text{temp}-\text{f}}$ in factory f
Compute the fitness of the factory f
End For
Sort all the factories according to the fitness
Update the Q value table according to the function
$Q(s_t, a_t) = (1 - \alpha)Q(s_t, a_t) + \alpha(r_{t+1} + \gamma maxQ(s_{t+1}, a_{t+1})$
Update the fitness value of each factory
Else
For f = 1 to F
Randomly generate a value p, $p \in \{0, 1\}$
If$p < 0.5$
Select the strategy with the largest Q value in the Q table and implement the corresponding action for the job sequence $\pi^{\text{temp}-\text{f}}$
Compute the fitness of the factory f
Else
Randomly select a strategy and implement it for the job sequence $\pi^{\text{temp}-\text{f}}$
Calculate the fitness value of the factory f
End For
Sort the factories according to the fitness
Update the Q value table according to the function
$Q(s_t, a_t) = (1 - \alpha)Q(s_t, a_t) + \alpha(r_{t+1} + \gamma maxQ(s_{t+1}, a_{t+1})$
Update the fitness of each factory
End If
End

E. The local search strategies

In this paper, five different local search strategies are designed to supersede the insertaion improvement strategy of the classic IG algorithm. Among them, there are two strategies for the swap of blocked jobs in the current factory. there are two strategies for the swap of all jobs in the current factory. The remaining one is to use the destruction-reconstruction strategy of traditional IG algorithm for the current factory. All of these strategies can disturb the blocked jobs, reducing the energy waste due to blocking constraints. The details of these strategies are shown in Algorithm 5.

Algorithm 5 The local search strategies

Input: the job sequence $\pi^{\text{temp}-\text{f}}$, action r, the number of the blocked jobs $count_{block}$, bool value flag
Output: $\pi^{\text{temp}-\text{f}}$
Begin
If r == 1 or r == 2
While flag == true
flag = false
For count = 1 to $count_{block}$

Swap any two blocked jobs in the $\pi^{\text{temp}-\text{f}}$, denoted the new sequence as $\pi_{new}^{\text{temp}-\text{f}}$
If $\pi_{new}^{\text{temp}-\text{f}}$ better than $\pi^{\text{temp}-\text{f}}$
If r == 1
$\pi^{\text{temp}-\text{f}} = \pi_{new}^{\text{temp}-\text{f}}$ flag = true
If r == 2
$\pi_{new}^{\text{interval}} = \pi_{new}^{\text{temp}-\text{f}}$, $\pi_{new}^{\text{temp}-\text{f}} = \pi^{\text{temp}-\text{f}}$
$= \pi^{\text{temp}-\text{f}} = \pi_{new}^{\text{interval}}$, flag = true
Else
$\pi_{new}^{\text{temp}-\text{f}} = \pi^{\text{temp}-\text{f}}$
End If
End For
End While
End If
If r == 3 or r == 4
While flag == true
flag = false
Forj = 1 to the number of the jobs in $\pi^{\text{temp}-\text{f}}$
$\pi^{\text{temp}-\text{f}'} = \pi^{\text{temp}-\text{f}}$
For i = 1 to the number of the jobs in $\pi^{\text{temp}-\text{f}}$
If $j \neq i$
Swap the $\pi_j^{\text{temp}-\text{f}'}$ and $\pi_i^{\text{temp}-\text{f}'}$, denoted the new sequence as $\pi_n ew^{\text{temp}-\text{f}'}$
End If
If $\pi_n ew^{\text{temp}-\text{f}'}$ better than $\pi^{\text{temp}-\text{f}'}$
If r == 3
Record the job position pos and energy consumption value minvalue
If r == 4
$\pi^{\text{temp}-\text{f}'} = \pi_n ew^{\text{temp}-\text{f}'}$
flag = true
End If
If r == 3
$\pi^{\text{temp}-\text{f}'} = \pi^{\text{temp}-\text{f}}$
End For
If r == 3
If minvalue ¡ the energy consumption of $\pi^{\text{temp}-\text{f}}$
Swap the $\pi_j^{\text{temp}-\text{f}}$ and $\pi_p os^{\text{temp}-\text{f}}$
flag = true
EndIf **End For** **End While**
If $\pi^{\text{temp}-\text{f}'}$ better than $\pi^{\text{temp}-\text{f}}$
$\pi^{\text{temp}-\text{f}} = \pi^{\text{temp}-\text{f}'}$
End If
If r == 5
The job sequence of the factory is destructed and reconstructed, the d value is a random number not greater than the number of the jobs, The specific steps can be found in references [5]
End If
End

IV. NUMERICAL RESULTS

A. Parameter settings

Different number of jobs, factories, and stages can combine different scale cases. This paper sets the total jobs as J, total factories as f and total stages as S, $J \in \{50, 100, 150, 200, 300\}$, $f \in \{2, 3, 4\}$ and $S \in \{5, 8, 10\}$. There are two identical parallel machines at each stage. For each $f \times J \times S$ scale problem, 10 instances are yield, thus, the number of experiment instances is$5 \times 3 \times 3 \times 10 = 450$. Processing times are produced uniformly distributed in interval [1, 30], the power consumption of idle, blocking and processing, are produced uniformly from the intervals [1, 2], [3, 4] and [5, 7], randomly.

To be fairness, we set same run time as the termination condition, denoted as TimeLimit.$TimeLimit = f \times J \times S \times CPU$,$CPU = 10$, All comparison algorithms are produced by C++, it runs in the Visual Studio 2019, 16GB mermory in Intel Core i7 Pentium processor with 2.60 GHZ. Every instance is tested 30 times.

B. Evaluation index

Because of the complexity of the calculation is very large, the best solution in DBHFSP is unknown, thus, we use the relative percentage deviation (RPD) [10] to analyse the performance of all algorithms. The equation is shown as follows.

$$RPD = (c_i - c_{best}/c_{best} \times 100) \tag{18}$$

where $c_{best}$is the minimum result gained by all comparison algorithms $c_i$is a value produces by method i. The algorithm that has the minimum RPD is greater than other comparison algorithms.

From the results of DBHFSP, we see that the specific result is too big to have a difference between the denominator and the numerator small. The RPD gained by all comparison algorithms are also very small. Thus, to comprehensively analyse the IGQ algorithm, we not only compare RPD values, but also compare the minimum energy consumption of all the algorithms.

C. The simulation experiments

We compare IGQ to 7 metaheuristics. The compared algorithms to solve the HFSP, GA [11], DABC [12], EMBO [13], DPSO [14], these algorithms use the same allocation factory strategy as the one used in this paper, in addition, there are also algorithms to solve DPFSP, i.e., CRO [15], IG [16], and the algorithm to solve the DHFSP, MN-IG [17], respectively. All methods have the same criteria of termination. As can be seen from TABLE I, the best results and RPD values of the compared methods are highlighted in bold.

For all different scale instances, TABLE I lists the best results gained by 7 methods. Besides, the RPD results of methods are shown in TABLE I. The IGQ gets the minimum value and the minimum RPD in all instances. It fully shows the superiority of the IGQ. We give the convergence charts of IGQ, CRO, IG, DPSO, EMBO, MN-IG algorithms in 2×100×10 scale and 4×200×5 scale, which are shown in Fig. 1 and Fig. 2, respectively. As can be seen from the Figures, the convergence curves of the IGQ are smooth, and the initial solution of

TABLE I
SIMULATION RESULTS COMPARED TO OTHER METHODS WHEN CPU = 10

		CRO		IG		DPSO		GA		EMBO		IGQ		MN-IG		DABC	
Instance	**J×S**	**best**	**RPD**	**best**	**RPD**	**best**	**RPD**	**best**	**RPD**	**best**	**RPD**	**best**	**RPD**	**best**	**RPD**	**best**	**RPD**
	50×5	27100	8.02	27533	9.75	26934	7.36	26053	3.85	26360	5.07	**25087**	**0**	25549	1.84	27548	9.8
	50×8	47904	7.61	47895	7.59	47327	6.32	45531	2.28	45958	3.24	**44513**	**0**	45596	2.43	48684	9.37
	50×10	55842	6.39	57766	10.06	55384	5.52	53981	2.85	54556	3.94	**52485**	**0**	53628	2.17	57400	9.36
	100×5	56967	7.17	57452	8.08	57818	8.77	56616	6.51	56962	7.16	**53155**	**0**	54579	2.67	58795	10.61
	100×8	96309	7.48	96362	7.54	96805	8.03	94561	5.53	95673	6.77	**89602**	**0**	91086	1.65	97367	8.66
	100×10	115305	6.8	117472	8.81	116131	7.57	114048	5.64	115141	6.65	**107954**	**0**	110338	2.2	116998	8.37
	150×5	82273	7.28	82817	7.99	84231	9.83	83174	8.46	83141	8.41	**76686**	**0**	78362	2.18	84962	10.79
	150×8	143264	7.45	144900	8.68	144271	8.21	142500	6.88	143391	7.55	**133319**	**0**	134908	1.19	145132	8.86
f=2	**150×10**	172720	7.26	175027	8.69	176196	9.42	173038	7.46	175432	8.94	**161021**	**0**	166392	3.33	177866	10.46
	200×5	103474	8.83	105276	10.72	105105	10.54	104213	9.61	104419	9.82	**95075**	**0**	96604	1.6	105657	11.13
	200×8	190084	7.56	191690	8.47	194018	9.79	191817	8.55	192566	8.97	**176708**	**0**	179387	1.51	195312	10.52
	200×10	227452	7.21	228427	7.67	229469	8.16	227701	7.33	228919	7.9	**212142**	**0**	215652	1.65	231601	9.17
	300×5	168692	8.23	170234	9.22	175189	12.4	173961	11.61	173977	11.62	**155856**	**0**	157573	1.1	175428	12.55
	300×8	275105	8.48	276250	8.93	281493	10.99	278680	9.89	281432	10.97	**253598**	**0**	257231	1.43	281593	11.03
	300×10	346114	7.6	348896	8.47	354094	10.08	352528	9.6	353461	9.89	**321641**	**0**	326359	1.46	353884	10.02
	mean	140573.67	7.56	141866.47	8.71	142964.33	8.87	141226.8	7.07	142092.53	7.79	**130589.47**	**0**	132882.93	1.89	143881.8	10.05
	50×5	30315	6.54	30932	8.71	30244	6.29	29436	3.45	29639	4.17	**28452**	**0**	29053	2.11	31093	9.28
	50×8	47355	5.35	48044	6.88	47548	5.78	46402	3.23	46911	4.36	**44949**	**0**	46746	3.99	49050	9.12
	50×10	59866	7.23	60767	8.84	59111	5.88	57596	3.16	58492	4.77	**55828**	**0**	57725	3.39	61573	10.29
	100×5	61952	5.65	62761	7.03	63069	7.56	62028	5.78	62572	6.71	**58636**	**0**	60072	2.44	64392	9.81
	100×8	90861	7.03	92453	8.91	92332	8.76	90332	6.41	91503	7.79	**84888**	**0**	87296	2.83	93693	10.37
	100×10	114935	7.44	116814	9.2	115954	8.4	113342	5.95	115021	7.52	**106968**	**0**	109464	2.33	117746	10.07
	150×5	83802	7.5	84621	8.55	85893	10.19	84626	8.56	85330	9.46	**77949**	**0**	80048	2.69	86503	10.97
	150×8	139326	7.65	140704	8.71	141858	9.61	139298	7.63	141165	9.07	**129419**	**0**	132546	2.41	142072	9.77
f=3	**150×10**	182628	6.85	184792	8.11	184852	8.15	182552	6.8	184691	8.06	**170914**	**0**	174641	2.18	186886	9.34
	200×5	108381	8.42	109123	9.16	110628	10.67	109534	9.58	109874	9.92	**99958**	**0**	101452	1.49	111287	11.33
	200×8	175798	8.36	178202	9.84	180046	10.98	178332	9.92	178954	10.31	**162226**	**0**	166083	2.37	181765	12.04
	200×10	240530	7.33	243285	8.56	244822	9.24	242531	8.22	243783	8.78	**224099**	**0**	229325	2.33	246082	9.8
	300×5	159096	6.89	159844	7.39	164568	10.57	163830	10.07	163780	10.04	**148832**	**0**	151071	1.5	164841	10.75
	300×8	273324	6.99	277199	8.51	279993	9.61	279245	9.31	279630	9.46	**255444**	**0**	258825	1.32	280987	9.99
	300×10	346952	6.92	352836	8.73	355125	9.44	353235	8.86	355273	9.48	**324485**	**0**	328093	1.11	355465	9.54
	mean	141008.07	7.08	142825.13	8.48	143736.2	8.74	142154.6	7.13	143107.87	7.99	**131536.47**	**0**	134162.67	2.3	144895.67	10.16
	50×5	27716	4.78	28115	6.29	27715	4.77	27187	2.78	27580	4.26	**26451**	**0**	28213	6.66	28803	8.89
	50×8	52526	7.73	52565	7.81	51761	6.16	50435	3.44	51073	4.75	**48757**	**0**	51160	4.92	53128	8.96
	50×10	67034	7.12	67178	7.35	66274	5.91	64643	3.3	65074	3.99	**62573**	**0**	65118	4.06	67241	7.46
	100×5	51386	5.81	53028	9.2	52844	8.82	51789	6.64	52658	8.43	**48560**	**0**	49895	2.74	53702	10.58
	100×8	89637	4.72	93585	9.33	92864	8.49	90519	5.75	92181	7.69	**85595**	**0**	88348	3.21	92940	8.58
	100×10	112875	5.34	114990	7.31	114712	7.05	112482	4.97	114231	6.6	**107152**	**0**	110617	3.23	115708	7.98
	150×5	82603	6.58	83163	7.3	84616	9.17	83599	7.86	84027	8.41	**77502**	**0**	79825	2.99	85710	10.59
	150×8	141051	77.69	143029	9.28	143992	10.01	141598	8.18	144092	10.09	**130882**	**0**	135155	3.26	145648	11.28
f=4	**150×10**	190064	6.9	192768	8.42	192463	8.24	189120	6.36	191803	7.87	**177795**	**0**	181841	2.27	194236	9.24
	200×5	108022	5.88	109908	7.73	112003	9.78	110636	8.44	111029	8.83	**102018**	**0**	103746	1.69	112094	9.87
	200×8	187361	6.59	188692	7.35	192623	9.58	189559	7.84	192898	9.74	**175771**	**0**	179221	1.96	193790	10.25
	200×10	229538	6.2	233131	7.86	236079	9.23	233314	7.95	235597	9	**216126**	**0**	220806	2.16	237844	10.04
	300×5	182590	6.98	185142	8.48	189360	10.95	188237	10.29	188950	10.71	**170665**	**0**	173490	1.65	189161	10.83
	300×8	287682	6.45	290146	7.36	296315	9.65	294100	8.83	296397	9.68	**270237**	**0**	275278	1.86	297048	9.92
	300×10	334820	7.1	340687	8.98	346698	10.9	343957	10.02	346228	10.75	**312606**	**0**	318926	2.02	347642	11.2
	mean	142993.67	11.06	145075.13	8	146687.93	8.58	144745	6.84	146254.53	8.05	**134179.33**	**0**	137442.6	2.98	147646.33	9.71

the proposed algorithm is better than other algorithms. With the increase of the time, the energy consumption of the IGQ algorithm is smaller than other algorithms. The reason for th result may the IGQ can extensively search the current solutic in the local neighborhood, and the global search strategy ca further increase the diversity of solutions in the later stage c the algorithm.

V. CONCLUSION

This paper first introduce the MILP model of DBHFS next, we propose a Q-learning method based on IG algc rithm, namely IGQ to optimize the energy consumption c job sequence. In IGQ, NEH combined with the global loca search strategy is designead to yield an initial solution, ther a global perturbation strategy and a Q-learning framewor are developed. The numerical results show the effective c the IGQ algorithm. Our future work would design mor and more strategies on the basis of IG algorithm. Late the multi-objective optimization of DBHFSP will be studiec In addition, we may research the assembly process or the batching machine problem, or make uncertainty restriction into this problem, such as the dynamic scheduling, machine breakdowns scheduling.

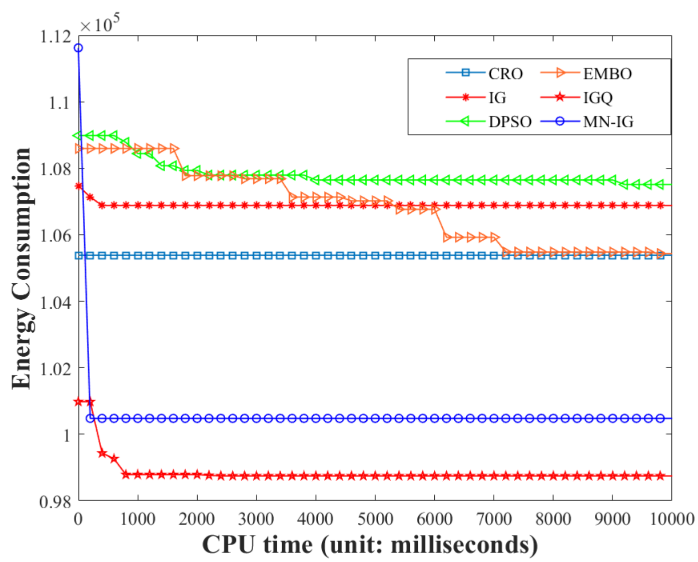

Fig. 1. The convergence curves for compared algorithms in scale (2×100×10)

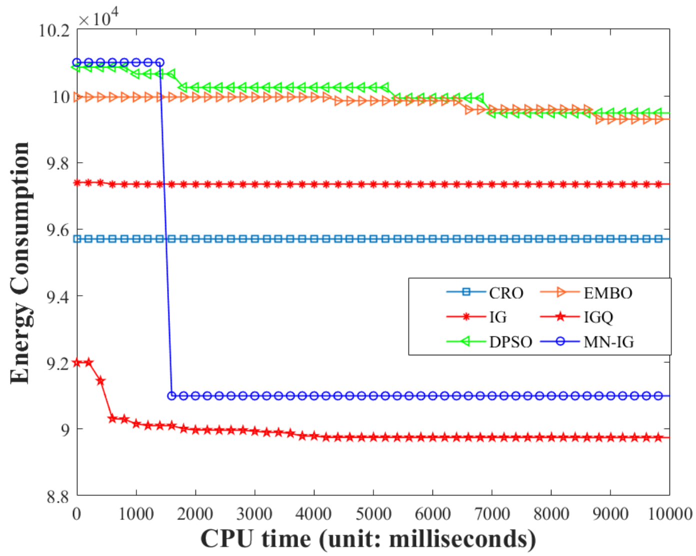

Fig. 2. The convergence curves for compared algorithms in scale (4×200×5)

Acknowledgment

The paper was supported by No. 61773246, 61603169, 61803192, 61966012, 61773192, 61973203, and 71533001 of National Natural Science Foundation of China.

References

[1] R. Ruiz, J. Antonio and V.Rodriguez, "The hybrid flow shop scheduling problem," Eur. J. Oper. Res. vol. 205, pp.1–18, 2010.

[2] B. Naderi and R. Ruiz, "The distributed permutation flowshop scheduling problem," Comput. Oper. Res. vol. 37, pp.754–768, 2010.

[3] H. J. H, J. Q. Li, Y. Du, M. X. Song, Y. Y. Zhang, "Solving distributed hybrid flowshop scheduling problems by a hybrid brain storm optimization algorithm," IEEE. Access. vol.99, pp.1–1, 2019.

[4] V. Caraffa, S. Ianes, T. P. Bagchi,C. Sriskandarajah, "Minimizing makespan in a blocking flowshop using genetic algorithms," Int. J. Prod. Econ. vol. 70, pp.101–115, 2001.

[5] R. Ruiz, T. Stutzle. "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem," Eur. J. Oper. Res. vol. 177, pp.2033–2049, 2007.

[6] R. Chen, B. Yang, S. Li, S. Wang. "A Self-Learning Genetic Algorithm based on Reinforcement Learning for Flexible Job-shop Scheduling Problem," Comput. Ind. Eng. vol. 149, 2020.

[7] J. J. Wang, L. Wang. "A Bi-Population Cooperative Memetic Algorithm for Distributed Hybrid Flow-Shop Scheduling," IEEE. Trans. Emerg. Topics. Comput. Intell. vol. 99, pp. 1–15, 2020.

[8] Muhammad. Nawaz. and, et al. "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem," Omega. vol. 11, pp. 91–95, 1983.

[9] J. P. Huang, Q. K. Pan, L. Gao. "An effective iterated greedy method for the distributed permutation flowshop scheduling problem with sequence-dependent setup times," Swarm. Evol. Comput. 2020, 59:1-18.

[10] H. Ztop, M. F. Tasgetiren, D. T. Eliiyi, Q. K. Pan, "Metaheuristic algorithms for the hybrid flowshop scheduling problem," Comput. Oper. Res. vol. 111, pp. 177–196, 2019.

[11] M. Nejati, I. Mahdavi, R. Hassanzadeh, N. M. Amiri, M. S. Mojarad, "Multi-job lot streaming to minimize the weighted completion time in a hybrid flow shop scheduling problem with work shift constraint," Int. J. Adv. Manuf. Technol. vol. 70, pp. 501–514, 2014.

[12] Q. K. Pan, L. Wang, J. Q. Li, J. H. Duan, "A novel discrete artificial bee colony algorithm for the hybrid flowshop scheduling problem with makespan minimisation," Omega. vol. 45, pp. 42-56, 2014.

[13] B.Zhang, Q. k. Pan, L. Gao, X. L. Zhang, H. Y. Sang, J. Q. Li, "An effective modified migrating birds optimization for hybrid flowshop scheduling problem with lot streaming," Appl. Soft. Comput. vol. 52, pp. 14-27, 2017.

[14] M. K. Marichelvam, M. Geetha, M. Tosun. "An improved particle swarm optimization algorithm to solve hybrid flowshop scheduling problems with the effect of human factors – a case study," Comput. Oper. Res. vol. 114, 2019.

[15] H. Bargaoui, D. O. Belkahla, G. Khaléd, "A novel chemical reaction optimization for the distributed permutation flowshop scheduling problem with makespan criterion," Comput. Ind. Eng. vol. 111, pp. 239–250 2017.

[16] R. Ruiz, Q.K. Pan, B. Naderi, "Iterated greedy methods for the distributed permutation flowshop scheduling problem," Omega. vol. 83, pp. 213–222, 2019.

[17] W. Shao, Z. Shao, D. Pi. "Modeling and multi-neighborhood iterated greedy algorithm for distributed hybrid flow shop scheduling problem," Knowl. Based. Syst. vol. 194, pp. 1–17, 2020.