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-factoy 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, 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.
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 several 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 constraints.

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 one machine.

3) All jobs should be continuously processed not be preempted 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:

1) \(J\): The number of jobs.

2) \(F\): The number of factories.

3) \(S\): The number of stages.

4) \(f_j\): The index of the jobs, \(j \in \{1, 2, ..., J\}\).

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

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

7) \(m\): The index of the machines at each stage, \(m \in \{1, 2\}\).

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

9) \(P_{f,s}\): Processing time of job \(j\) at stage \(s\).

10) \(EC^\text{Process}_{f,s}\): Energy consumption per unit time of a job which is processed at stage \(s\) in factory \(f\).

11) \(EC^\text{Blocking}_{f,s}\): Energy consumption per unit time of a job which is blocked at stage \(s\) in factory \(f\).

\(EC^\text{Idle}_{f,s}\): 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.

\(x_{f,s,j}\): 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:

\[\min TEC = PEC + BEC + IEC\]  

\[PEC = \sum_{f=1}^{F} \sum_{s=1}^{S} \sum_{j=1}^{J} EC^\text{Process}_{f,s} \cdot p_{j,s} \cdot y_{f,s,j,m}\]  

\[m = 1||m = 2\]  

\[BEC = \sum_{f=1}^{F} \sum_{s=2}^{S} \sum_{j=1}^{J} EC^\text{Blocking}_{f,s} \cdot (m_{f,s} - C_{f,j,s} - 1) \cdot x_{f,j}\]  

\[m_{f,s} \geq C_{f,j,s} - 1\]  

\[IEC = \sum_{f=1}^{F} \sum_{s=2}^{S} \sum_{j=1}^{J} EC^\text{Idle}_{f,s} \cdot (C_{f,j,s} - m_{f,s}) \cdot x_{f,j}\]  

\[m_{f,s} \leq C_{f,j,s}\]  

\[s.t. \sum_{f=1}^{F} x_{f,j} = 1, \forall j\]  

\[\sum_{m=1}^{2} y_{f,s,j,m} = x_{f,j}, \forall f, j, s\]  

\[B_{f,j,1} \geq 0, \forall f, j\]  

\[B_{f,j,s+1} \geq B_{f,j,s} + p_{j,s} \cdot y_{f,j,s},\forall f, j, s\]  

\[z_{f,s,j,j'} + z_{f,s',j',j} \leq 1, \forall f, j, s, j, j'\]  

\[z_{f,s,j,j'} + z_{f,s',j',j} \geq y_{f,s,j,m} + y_{f,s,j',m} - 1, \forall f, s, j > j', m = 1||m = 2\]
The framework of IGQ is given in Algorithm 1.

**Algorithm 1** The framework of the IGQ Algorithm

- **Input:** \( \pi = \{ \pi_1, \pi_2, \ldots, \pi_J \} \) all parameters used in this algorithm
- **Output:** \( \pi_{\text{best}} \) and TEC
- **Begin:**
  \( \pi_{\text{temp}} = \pi \)
- **Initialization:** Using the NEH to assign the jobs to the F factories.
  \( \text{GlobalSearchStrategy}(\pi, \pi_{\text{temp}}); \)
- While the termination criterion is not satisfied do
  - local search strategy:
    - \( \text{Q-learning-method}(\pi_{\text{temp}}); \)
  - global search strategy:
    - \( \text{GlobalSearchStrategy}(\pi, \pi_{\text{temp}}); \)
    - If \( \pi_{\text{temp}} \) better than \( \pi \) then
      \( \pi = \pi_{\text{temp}} \)
      **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:** \( \pi = \{ \pi_1, \pi_2, \ldots, \pi_J \} \)
- **Output:** \( \pi_{\text{temp}} \)
- **Begin:**
  - Computing the total processing time of all jobs
  - Sorting these jobs in a descending order, denoted as \( \pi_{\text{temp}} \)
  - For \( j=1 \) to \( F \)
    - Take job \( \pi^j_{\text{temp}} \) from \( \pi_{\text{temp}} \) and arrange to factory \( j \)
  **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:** \( \pi, \pi_{\text{temp}}, \) bool value \( \text{flag} \)
- **Output:** \( \pi_{\text{temp}} \)
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 is 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 $Q(s_t, a_t) = (1 - \alpha)Q(s_t, a_t) + \alpha (r_{t+1} + \gamma \max Q(s_{t+1}, a_{t+1})$. The $Q(s_t, a_t)$ represents the Q value that take the action $a_t$ at state $s_t$, $\alpha$ shows the learning rate, $\gamma$ 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: $\pi^{temp}$
Output: $\pi^{temp}$

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

E. The local search strategies

In this paper, five different local search strategies are designed to supersede the insertion 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 $\pi^{temp} - f$, action $r$, the number of the blocked jobs $count_{block}$, bool value flag
Output: $\pi^{temp} - f$

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 \( \pi^{\text{temp-f}} \), denoted
the new sequence as \( \pi^{\text{new}} \).

If \( \pi^{\text{temp-f}} \) better than \( \pi^{\text{new}} \)

\[ \text{If } r = 1 \quad \pi^{\text{temp-f}} = \pi^{\text{temp-f}} \quad \text{flag} = \text{true} \]

\[ \text{If } r = 2 \quad \pi^{\text{interval}} = \pi^{\text{new}} \quad \pi^{\text{new}} = \pi^{\text{temp-f}} \quad \pi^{\text{temp-f}} = \pi^{\text{interval}} \quad \text{flag} = \text{true} \]

Else
\[ \pi^{\text{temp-f}} = \pi^{\text{temp-f}} \]
End If
End For
End While
End If
If \( r = 3 \) or \( r = 4 \)
While flag == true
flag = false
For \( i = 1 \) to the number of the jobs in \( \pi^{\text{temp-f}} \)
\[ \pi^{\text{interval}} = \pi^{\text{new}} \quad \pi^{\text{new}} = \pi^{\text{temp-f}} \quad \pi^{\text{temp-f}} = \pi^{\text{interval}} \quad \text{flag} = \text{true} \]
End If
End For
End While
If \( r = 3 \)
Record the job position pos and energy consumption value minvalue
\[ \text{If } r = 4 \quad \pi^{\text{temp-f'}} = \pi^{\text{new}} \quad \pi^{\text{temp-f'}} = \pi^{\text{temp-f'}} \quad \text{flag} = \text{true} \]
End If
If \( r = 3 \)
\[ \pi^{\text{temp-f'}} = \pi^{\text{temp-f}} \]
End For
If \( r = 3 \)
If minvalue \( j \) the energy consumption of \( \pi^{\text{temp-f}} \)
Swap the \( \pi^{\text{temp-f'}} \) and \( \pi^{\text{temp-f'}} \), denoted the
new sequence as \( \pi^{\text{new}} \).
End If
End For
End While
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 com-
bine different scale cases. This paper sets the total jobs
as \( J \), total factories as \( f \) and total stages as \( S \), \( J \in \{50, 100, 150, 200, 3000\} \), \( f \in \{2, 3, 4\} \) 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. The \( \text{TimeLimit} = f \times J \times S \times \text{CPU}, \text{CPU} = 10 \). All comparison algorithms are produced by
C++, it runs in the Visual Studio 2019, 16GB memory 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.

\[ RPD = \left( \frac{c_i - c_{\text{best}}}{c_{\text{best}}} \times 100 \right) \quad (18) \]

where \( c_{\text{best}} \) 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 algo-
rithms 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 DPFSF, i.e., CRO [15], IG [16], and
the algorithm to solve the DHFSF, 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 \( 2 \times 100 \times 10 \)
scale and \( 4 \times 200 \times 5 \) 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
the proposed algorithm is better than other algorithms. With the increase of the time, the energy consumption of the IG algorithm is smaller than the other algorithms. The reason for this result may be that the IGQ can extend the search time in the local neighborhood, and the global search strategy can further increase the diversity of solutions in the later stage of the algorithm.

V. CONCLUSION

This paper first introduce the MILP model of DBHFS next, we propose a Q-learning method based on IG algorithm, namely IGQ to optimize the energy consumption in the job sequence. In IGQ, NEH combined with the global loc-search strategy is designed to yield an initial solution, then a global perturbation strategy and a Q-learning framework are developed. The numerical results show the effectiveness of the IGQ algorithm. Our future work will design more and strategies on the basis of IG algorithm. Later the multi-objective optimization of DBHFS will be studied. 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. 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